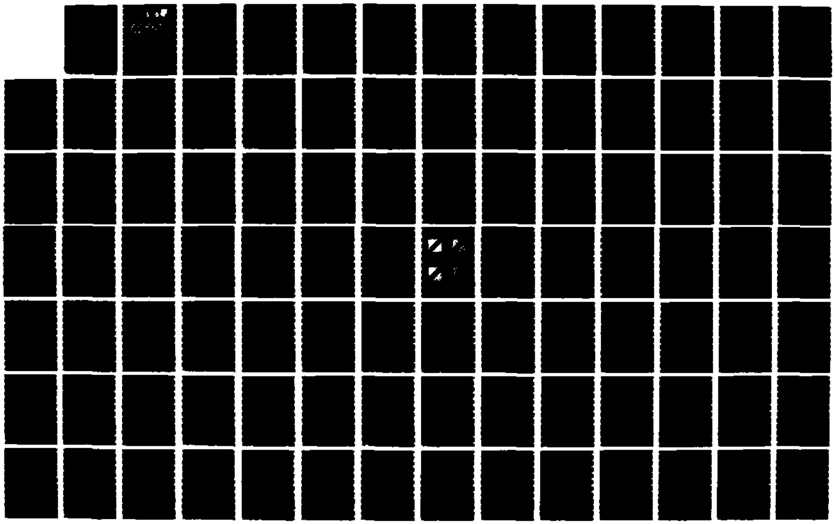
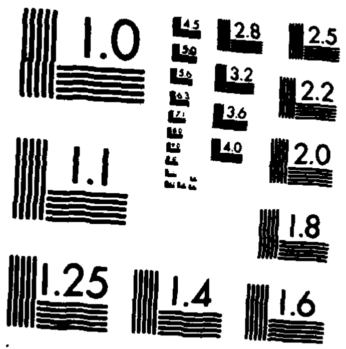
AD-A173 600 ADVANCED ARCHITECTURES FOR SIGNAL PROCESSORS (A SIMPLE 1/2
RNS (RESIDUE NUMB (U) MITRE CORP BEDFORD MA
D O CARHOUN ET AL JUL 86 MTR-9727 RADC-TR-86-59
UNCLASSIFIED F19628-84-C-0001 F/G 9/2 NL




1.0
4.5
5.0
5.6
6.3
7.1
8.0
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

(12)

AD-A173 600

RADC-TR-86-59
Final Technical Report
July 1986

ADVANCED ARCHITECTURES FOR SIGNAL PROCESSORS: (A SIMPLE RNS/SYSTOLIC ARCHITECTURE FOR ISOLATED WORD RECOGNITION)

The MITRE Corporation

D. O. Carhoun, W. L. Eastman and A. L. Bequillard

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700

86 11 4 020

RADC-TR-86-59 has been reviewed and is approved for publication.

APPROVED: Z. J. Pryk

ZENON J. PRYK
Project Engineer

APPROVED: [illegible]

FRANK J. REHM
Technical Director
Surveillance Division

FOR THE COMMANDER: John A. Ritz

JOHN A. RITZ
Plans and Programs Division

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

ADA 173600

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS: N/A
2a. SECURITY CLASSIFICATION AUTHORITY: N/A	3 DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S): MTR 9727	5 MONITORING ORGANIZATION REPORT NUMBER(S): RADC-TR-86-59
6a. NAME OF PERFORMING ORGANIZATION: The MITRE Corporation; 6b OFFICE SYMBOL (If applicable)	7a NAME OF MONITORING ORGANIZATION: Rome Air Development Center (OCTS)
6c. ADDRESS (City, State, and ZIP Code): Bedford MA 01731	7b ADDRESS (City, State, and ZIP Code): Griffiss AFB NY 13441-5700
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: Electronic Systems Division; 8b. OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: F19628-84-C-0001
8c. ADDRESS (City, State, and ZIP Code): Hanscom AFB MA 01731-5000	10 SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.
N/A	MOIE	74	40

11 TITLE (Include Security Classification) ADVANCED ARCHITECTURES FOR SIGNAL PROCESSORS: (A SIMPLE RNS/SYSTOLIC ARCHITECTURE FOR ISOLATED WORD RECOGNITION)

12 PERSONAL AUTHOR(S) D.O. Carhoun, W.L. Eastman, A.L. Bequillard

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Final	FROM Oct 84 TO Oct 85	July 1986	174

16 SUPPLEMENTARY NOTATION
N/A

17 COSATI CODES

FIELD	GROUP	SUB-GROUP
09	03	
15	04	

18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Word Recognition
Signal Processors
Systolic Architecture

19 ABSTRACT (Continue on reverse if necessary and identify by block number)
Isolated work recognition is a computationally intensive processing function for which the combination of residue number system (RNS) computation and systolic array architecture offers practical simplification in the design of a special-purpose hardware processor. We have developed a simple architecture that uses short-time correlation analysis to form the spectral patterns, a distortion function employing squared Euclidean distance between the normalized correlation values of the test and reference utternace segments, and a two-dimensional pipelined processor array to implement dynamic time-wraping for pattern registration. With the exception of the normalization of the sample correlation values, all significant computations are carried out in a residue number system of compact size to be implemented in a specially designed hardware of low complexity. This combination of techniques provides for a very high processing throughput in simple hardware that can be used for real-time word recognition in a large vocabulary.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT: ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL: Pryk, Zenon J.	22b TELEPHONE (Include Area Code): (315) 330-4437; 22c OFFICE SYMBOL: RADC (OCTS)

DD FORM 1473, 84 MAR   83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

PREFACE

This report is the second of two reports concerned with the development and application of improved techniques of digital signal processing based on the use of residue number systems and systolic array architectures to implement the processing functions associated with isolated-word speech recognition. It constitutes final documentation, for fiscal year 1985, on MITRE Mission Oriented Investigation and Experimentation (MOIE) Project 7440: Advanced Architectures for Signal Processors. The work was sponsored by the Rome Air Development Center, RADC/OCTS, under contract F19628-84-C-0001.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION 1

INTRODUCTION AND BACKGROUND

1.1 INTRODUCTION

The Advanced Architectures for Signal Processors project continues to apply residue number system (RNS) techniques to the practical design of advanced digital signal processors. We are developing algorithms, computational techniques, and systolic architectures to be implemented in custom-design, very large scale integrated (VLSI) electronics.

In conjunction with MITRE's Integrated Electronics and Mathematical Research projects, we have been exploring opportunities for improved implementation of digital signal processing functions fostered by VLSI hardware design combining RNS computation with systolic architectures, i.e., arrays of identical pipelined processors using nearest-neighbor communication.

1.2 BACKGROUND

In FY84 and continuing into FY85, we explored the use of RNS for implementing the computationally intensive processing functions associated with speech recognition. Algorithms for word recognition require three essential processes: (1) generation of a test pattern, or spectrogram, to efficiently describe an utterance; (2) computation of a measure of the distortion between segments of spoken and referenced utterances; and (3) dynamic programming to effect registration between utterances that differ because of local time expansions and contractions, i.e., dynamic time-warping (DTW).

These processing functions can impose an enormous burden for a processor employing a large reference vocabulary (several hundred words). If performed sequentially, conventional signal processing cannot achieve real-time operation for normal speaking rates of a few words per second. The combination of RNS and systolic architectures can reduce the otherwise large hardware complexity and increase the throughput sufficiently enough to achieve real-time recognition rates.

Residue number systems are well suited for implementing linear processes composed of multiplication and addition in low-complexity hardware, especially when the computational result can be confined to a small range. On the other hand, RNS is usually unsuitable for nonlinear operations, and division or magnitude comparison generally requires reconversion to a weighted number system.

In speech recognition, the distortion computations, largely multiply-and-accumulate operations dependent on the chosen measure, must be performed for every compared pair of test and reference segments. RNS is useful for this situation. The dynamic programming step, however, requires local magnitude comparisons for the accumulated distortion of a (conceptual) least-distortion path traversing an array of distortion values that compare different time segments of a test and reference utterance pair. Magnitude comparisons are awkward for RNS, but if the cumulative local path differences are small enough they can be contained within the range of a single modulus. The entire RNS is used to accumulate the total path cost for comparison with path costs of other utterance pairs. Occasional local overflow of the chosen modulus probably does not harm the path-cost computation.

Confining the distortion computation's dynamic range is necessary to successfully employ RNS for DTW. Because an RNS needs no allowance for local overflow of summed products, it has a dynamic range advantage over conventional multiply-and-accumulate computation representations. However, the gross dynamic range of the distortion computation must be confined to maintain realistic hardware complexity for parallel, pipelined operation in a few VLSI circuits.

1.2.1 Previous Results

In FY84, we found RNS techniques effective for computations used in word recognition processors. Using computer simulation, we developed a design model for a word recognition system employing autocorrelation analysis of segmented speech to calculate linear predictive coding (LPC) distortion-value inputs to a dynamic-programming algorithm for nonlinear time-warping. We identified systolic architectures for the key processing steps. Incorporating distortion computations with path-metric computations in a pipelined two-dimensional systolic array provided a high throughput for the most computationally intensive part of the recognition algorithm. The results were documented in an earlier project report [1].

While we were convinced by our results that RNS implementation had high potential for speech recognition, we were also convinced that substantial reduction of the gross dynamic range of the distortion computations (without sacrificing discrimination ability) would be required before RNS implementation could be accepted as a practical alternative to conventional architectures. This mandated a reassessment of the distortion function supporting the DTW computations. A particularly perplexing fact was that the path-metric computations could be successfully carried out with coarse quantization of the distortion values while computing our RNS implementation of the LPC distortion metric required a much larger range, even though RNS reduced the range requirement somewhat.

1.2.2 Distortion Function Alternatives

From our point of view, not only must the distortion measure produce satisfactory discrimination in a narrow range of values, it also must be suitable for RNS computations. For speech recognition, the purpose of the spectral analysis and distortion computation is to distill the information contained in the speech waveform into a small set of data suitable for low-error discrimination between distinct word patterns, and not necessarily to preserve information needed for high-grade speech synthesis. Thus, the LPC methods could be unnecessarily stringent for speech recognition, while admittedly weak in the presence of noise, and at the same time could impose the need for high-precision, i.e., large dynamic-range, computations.

Contemporary speech research rejects traditional squared Euclidean distance or equivalently mean-squared error as a distortion measure on the grounds that it is not sufficiently meaningful to represent what are considered requirements of auditory perception. The ear needs only to recognize the random process producing a waveform to within some degree of accuracy and does not need to have an accurate reproduction of the specific waveform. Demanding a small mean-squared error in a speech system requires more bits and accuracy than the human ear requires for intelligible speech.

The success of LPC methods can be attributed to the corresponding distortion measures (such as maximum likelihood) that assess, in a probabilistic sense, the similarity between original and reproduced processes or models rather than the actual waveforms. In our previous work, we accepted this rationale.

Mean-squared error, however, cannot be rejected on the grounds that it is too forgiving. We realized that if we were to base our analysis on power spectra or on autocorrelation analysis, then Euclidean distance would still be useful to discriminate spectral patterns for automatic word recognition. For RNS implementation, we preferred squared Euclidean distance, because it avoids explicit sign detection that would require exiting RNS.

The square-law behavior of the distortion function at first seemed an obstacle to reducing the dynamic range, but we realized we could quantize the distortion values to just a few levels and still maintain a good discrimination ability. This suggested that we could scale down the input values to keep within a practical range. We considered two methods of using squared Euclidean distance for RNS distortion computation, log spectral deviation and direct autocorrelation analysis.

Log Spectral Deviation

One of the oldest distortion measures proposed for speech is the L_p norm of the difference of the logarithms of the power spectra. Assuming the spectral envelopes have been sampled and scaled logarithmically, the L_2 norm is simply the square root of the squared Euclidean distance between the vectors of logarithmic spectral samples. One of the traditional ways to provide the spectral envelopes is through a bank of constant-Q filters appropriately spaced across the speech spectrum. The output of each filter's power is sampled in time and scaled logarithmically.

Such a filter bank probably is best implemented with analog-sampled, switched-capacitor active filters rather than with digital filters. Thus, we did not consider using RNS for the filter bank but would convert the log-spectral samples to RNS code for computation of the squared Euclidean distortion in a systolic array.

Another means considered for performing the filter-bank analysis was to take a discrete Fourier transform (DFT) of the windowed speech samples with a moderately high resolution (perhaps 256 to 1024 samples per frame), subdivide the samples into appropriate bands, compute the power in each subband, and convert to logarithmic form. With the exception of logarithmic conversion, all of the processing could be carried out with RNS.

Logarithmic conversion requires reconversion to a weighted number system (probably needed anyway for spectral normalization) followed by reconversion to RNS for the distortion computation. These processing steps are more complicated than the computation of the autocorrelation samples in the already-developed linear systolic array architecture. The operations, however, are performed only once for each test segment.

<u>Direct Correlation Analysis</u>

Alternatively, the autocorrelation coefficients of the windowed speech samples could be transformed by a DFT to provide a representation of the spectral envelope. We thought, however, that it might be better to use the coefficients directly for spectral discrimination. While LPC analysis approximates the spectral envelope as represented by the all-pole linear filter model, it is essentially a linear transformation of a subset of autocorrelation coefficients.

The autoregressive nature of the all-pole model allows this subset of autocorrelation values to approximate the remaining values. This is the basis for obtaining a good spectral approximation.

If the all-pole model is adequate for speech, then it should be satisfactory for automatic word recognition to employ directly the subset of autocorrelation coefficients used in LPC analysis in a squared Euclidean distance computation. Since it would be appropriate to work with normalized correlation coefficients for RNS implementation, it would be necessary to exit RNS to carry out the normalization and then reconvert to RNS for the distance computations. Such a technique for distortion computation is simpler to implement than either the log spectral deviation or LPC distortion, and there is no need to solve the normal equations for construction of the reference library as in the LPC method.

1.3 SUMMARY OF NEW RESULTS

In FY85 we examined experimentally the direct autocorrelation method by computer simulation using an expanded data base. We concluded that it has the desired effect of dramatically reducing the computation's range. Our simulations show a reduction from an equivalent of 30-bit distortion samples to about 9 bits.

At the same time, we found that the reduced range can be used to reduce the quantization of the input speech samples, and this suggests an additive noise tolerance that we would not expect for the LPC method. We studied the implications of the Euclidean distortion metric on the RNS systolic array implementation of the DTW algorithm and found considerable opportunity for hardware simplification resulting from a reduced range and a simpler algorithm.

We also developed a simplified method for quantizing the RNS representation of distortion values to a smaller range using a partial mixed-radix conversion that establishes natural quantization boundaries. These simplifications permit the distortion computations and subsequent quantization to be performed easily in the same pipelined array used for DTW.

The net result of our work is confidence in an RNS implementation of an effective word recognition algorithm in a systolic architecture. This architecture is of low complexity and high throughput, and is a strong candidate for VLSI implementation.

The processing system studied and simulated is illustrated by the block diagram of figure 1.1. The first processor block applies some preprocessing functions to the digitized test speech, but not in RNS. The autocorrelation analysis is performed in RNS, using a modified version of a linear systolic array previously designed for transversal filtering by the staff of MITRE's Integrated Electronics project. The output of this processing block is a set of normalized correlation samples. The normalization is carried out by a of reference words. Most of the processing occurs in the DTW processing block. Included in this processor are the Euclidean distance distortion computations, mixed-radix quantization, and dynamic programming computations needed to determine the minimum cost path.

IA-73 100

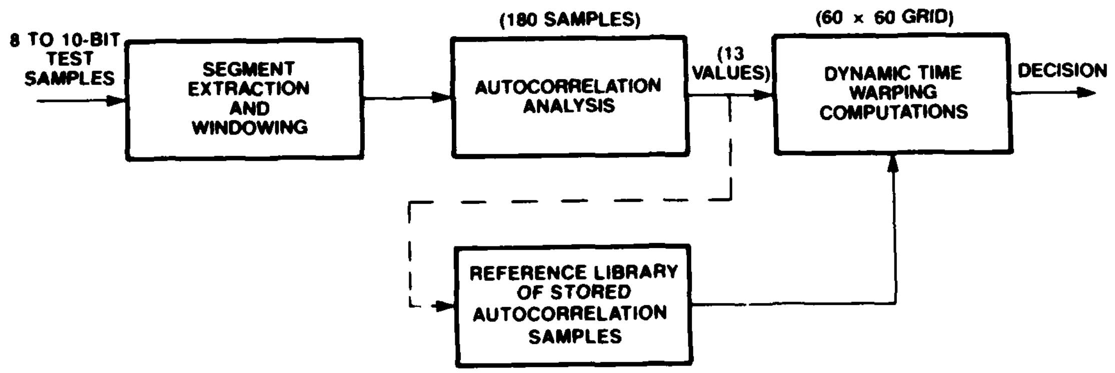

8 TO 10-BIT TEST SAMPLES
SEGMENT EXTRACTION AND WINDOWING
(180 SAMPLES)
AUTOCORRELATION ANALYSIS
(13 VALUES)
(60 × 60 GRID)
DYNAMIC TIME WARPING COMPUTATIONS
DECISION
REFERENCE LIBRARY OF STORED AUTOCORRELATION SAMPLES

Figure 1.1. RNS-Based Word Recognition

1.4 SCOPE

Section 2 of the report discusses the various processing and algorithmic steps, with RNS computation assumed. Section 3 presents the results of simulation of the RNS algorithm. Section 4 discusses an RNS implementation in a systolic array architecture. The results are summarized in section 5.

SECTION 2

PROCESSING FUNCTIONS FOR AN RNS WORD RECOGNITION SYSTEM

Figure 2.1 is a schematic diagram of a DTW-based word recognition system [2,3]. A detected test input utterance (a word to be matched to one contained in a reference library of stored utterances) is analyzed in short blocks of overlapping segments or frames. From each frame a vector of autocorrelation coefficients is computed. Segmentation into short blocks allows the process to be viewed as locally stationary, the time variation being accommodated by the sequential processing of overlapping analysis segments. It is assumed that a similar analysis has been performed on the utterances contained in the reference library. Autocorrelation vectors are compared to produce a local measure of distortion between individual segments of the test utterance and those of one of the reference utterances.

If there are n segments of the test utterance and m segments of the reference utterance, then the local distortions define a two-dimensional grid of $n \times m$ distortion values based on an appropriate distance metric. The low values correspond to good matches between analysis segments, and the high values correspond to poor matches. The purpose of the DTW algorithm [2,3] is to effect time registration between the stored reference and test segments to compensate for local time expansion or contraction of the test utterance with respect to the reference. It is a dynamic programming algorithm that calculates the accumulated weighted distortions for the least-cost path through the grid of distortion values. This score for the comparison of utterances is normalized and then compared in magnitude with the normalized scores for other pairings to produce a final decision.

IA-73 010

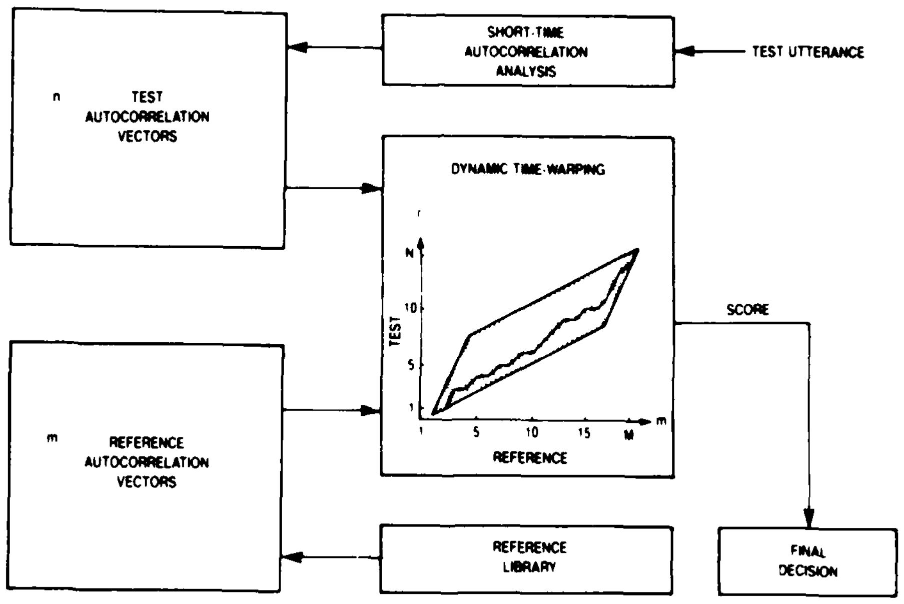


Figure 2.1. Speech Recognition via Dynamic Time-Warping

The local distortion computations, together with the path computations, impose the greatest computational burden in the recognition process. A local distortion must be computed for each selected pair of test and reference frames. These computations must be repeated for each reference utterance stored in the reference library and used in the shortest-path computations to produce a set of DTW scores from which the best match is determined. This computational bottleneck in the word recognition process can be impacted by a combination of RNS computation and systolic array architecture.

2.1 PREPROCESSING OF THE SPEECH WAVEFORM

The word recognition process involves digital computations on overlapping segments of the analog waveform. Preprocessing of the speech signals is required to obtain the appropriate sampled (8 kHz) digitized (16-bit) signals. Input processing of the speech waveforms and their A/D conversion are pictured in fiqure 2.2. Voice signals are picked up by the microphone, amplified, and passed through low-pass filters to remove frequency components above 4 kHz. After equalization to compensate for a finite sampling aperture, the analog signals are converted to 16-bit digital samples at an 8 kHz sampling rate by an A/D converter. Output from the A/D converter is either stored in a designated file for future input, or, in recognition mode, input directly to the word recognition system.

IA-72,365

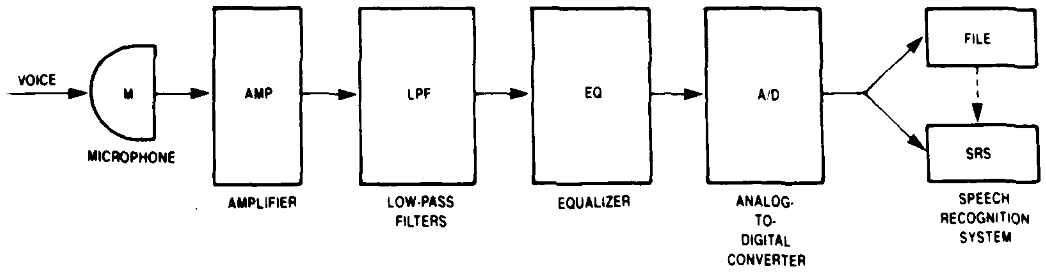


Figure 2.2. Input Speech Processing

2.2 UTTERANCE DETECTION AND INPUT QUANTIZATION

Detection of an utterance, as contrasted with a period of silence, is regarded as a digital preprocessing function. In our experimentation, we have based utterance detection on observation of energy statistics. The procedure is pictured in figure 2.3. The energy statistic is a measure of the short-time average signal energy minus the long-time (exponentially averaged) signal energy.

Three thresholds are set as shown in the figure. The beginning of an utterance is detected if the energy statistic rises above the START threshold and remains above it until crossing the HIGH threshold. The end of an utterance is detected if the energy statistic falls below the END threshold and remains below it for at least 150 ms. These events constitute a valid utterance detection if the length from beginning to end is at least 240 ms.

IA-72.368

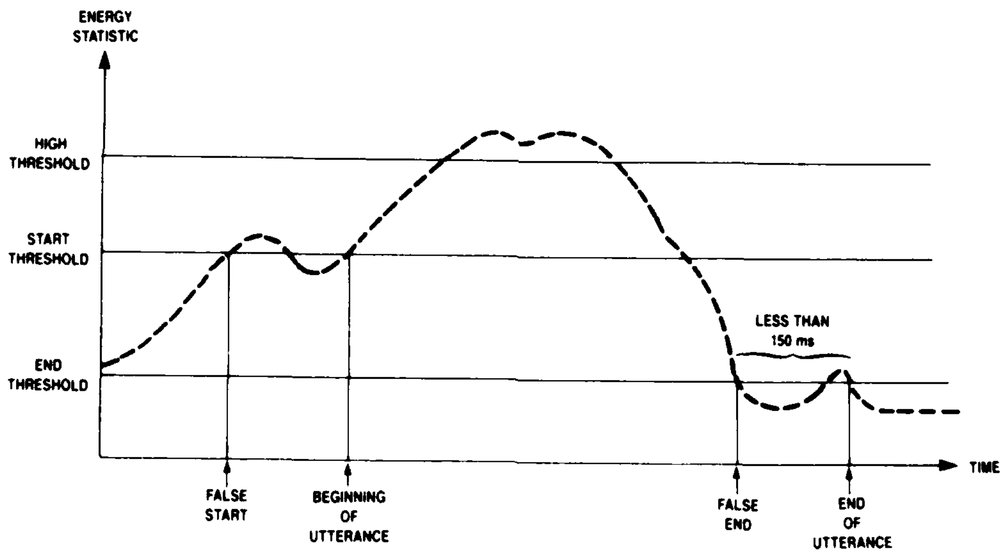


Figure 2.3. Utterance Detection

The digitized output of the A/D converter (figure 2.2) consists of 16-bit samples. Since 16 bits is more than is needed for isolated word recognition, the input samples are quantized to a smaller number of bits, usually 8 or 10, by truncation of the low-order bits of the sample. Performance results for various input quantizations from 2 to 16 bits are reported in section 3.4.

2.3 PRE-EMPHASIS AND NORMALIZATION OF SPEECH SAMPLES

The N digitized samples of a detected utterance in the speech input stream are normalized prior to analysis to render the analysis insensitive to system gain variations. Prior to normalization, however, pre-emphasis of the speech samples may be introduced if desired. The first-order predictor pre-emphasis filter employed in our experimental work is shown schematically in figure 2.4. The first sample of the utterance is unchanged; each remaining sample of the utterance is modified as shown by subtracting from it a constant μ times the preceding sample, where we set μ to the value 0.95. Investigations of the effects of pre-emphasis on word recognizability are reported in section 3.3.

Normalization of the speech samples is accomplished by multiplying each sample by a constant related to the input quantization and dividing the result by the RMS value of the N samples of the utterance. This result is then clipped, if necessary, so that the final integer value lies within the range specified by the chosen input quantization. The constant employed in the premultiplication of the samples is selected so that clipping occurs occasionally but not frequently.

IA-72 915

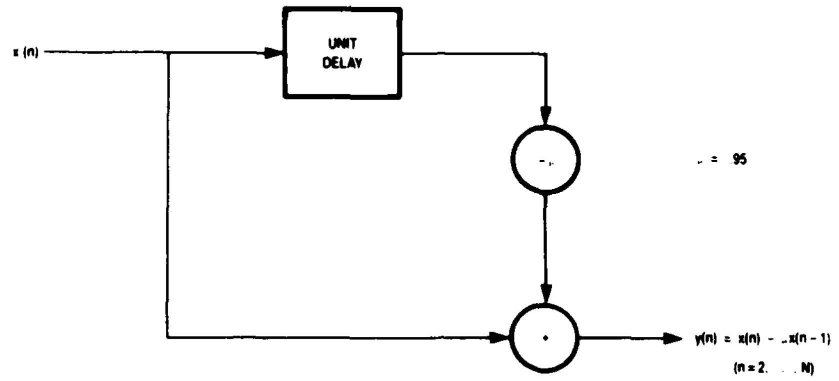


Figure 2.4. Pre-Emphasis of Speech Samples

2.4 AUTOCORRELATION ANALYSIS OF UTTERANCES

The purpose of spectral analysis in a DTW-based word recognition system is to facilitate the evaluation of a local distortion measure or distance function between selected pairs of reference frame data and test frame data. In our earlier work [1] we chose a method of spectral analysis based upon LPC employing variants of the Itakura-Saito distortion measure. This distortion function takes the form

$$d_{IS} = \text{scalar} + \sum_{j} r_j u_j \qquad (1)$$

where the r_j represent the calculated autocorrelation coefficients for the test frame data, a short segment of the test utterance data, and the u_j represent the inverse correlation coefficients for a stored reference frame [1]. Unfortunately, the u_j are typically small fractional values that must be scaled up before conversion to integers for RNS computation. Although the form of equation (1) appears to be well-suited for RNS implementation, the required scaling introduces difficulties. From our analysis [1] we expected to require an RNS range of about 30 bits to contain this computation. Simulation results for three RNS with respective ranges of approximately 30, 29, and 28 bits showed corresponding recognition error rates of 1.7%, 40%, and 100%. These results were particularly perplexing in the light of our success in performing the path computations using a coarse quantization of the distortion function values. It did not seem reasonable to have to employ so many bits to perform the distortion computation if almost all the precision in the outcome was discarded in the coarse quantization employed for the path computations.

In FY85, we replaced the LPC analysis with a simpler autocorrelation analysis based on the use of a squared Euclidean distance metric (section 2.6). While the square-law behavior of this distortion function poses some dynamic range problems, the realization from our previous work that we can maintain good discrimination between similar and different words while quantizing the computed distortion values to just two (or a few) levels suggests input quantization as a means for restricting the RNS computations to a practical range.

The correlation coefficients employed are those previously calculated and employed for LPC analysis of the speech utterances. Following detection, an utterance is divided into segments, or frames, for short-time analysis. The segmentation used in our experimental work is shown schematically in figure 2.5. We segment each utterance into overlapping frames of 22.5 ms. of speech, with an advance of 10 ms (overlap of 12.5 ms). Thus, each frame consists of 180 samples (at an 8 kHz sample rate), with each frame advanced by 80 samples over its predecessor.

For each frame of 180 samples we compute the P + 1 autocorrelation coefficients defined as

$$r_j = \sum_{i=0}^{179-j} \hat{x}_i \hat{x}_{i+j} \qquad (j = 0, 1, \ldots, P) \qquad (2)$$

where the components $\hat{x}_i$ of the L-th windowed segment are defined by

$$\hat{x}_i = w_i x_{80L+i} \qquad (i = 0, 1, \ldots, 179) \qquad (3)$$

IA-72,367

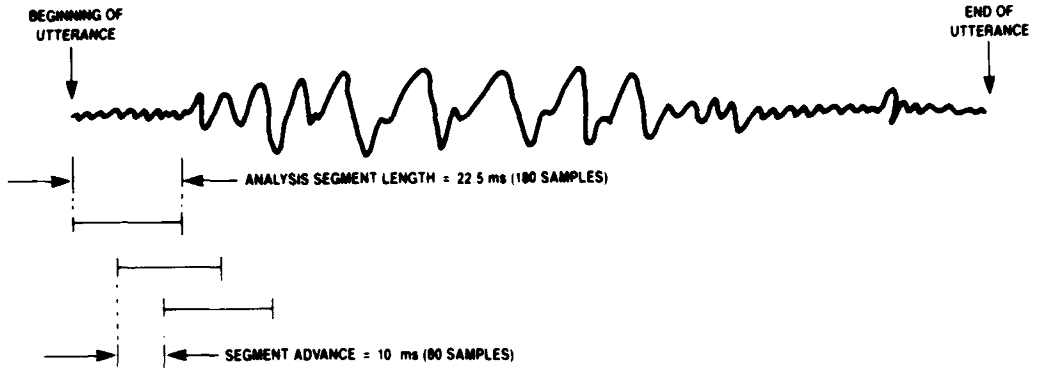

BEGINNING OF UTTERANCE
END OF UTTERANCE
ANALYSIS SEGMENT LENGTH = 22.5 ms (180 SAMPLES)
SEGMENT ADVANCE = 10 ms (80 SAMPLES)

Figure 2.5. Segment Extraction and Windowing

and the w_i are weights that depend upon the type of window used. In all our work we employed a Hamming window. We generally chose $P = 12$, but the effects of making other choices are reported in section 3.5.

The same analysis is applied to both test and reference utterances. The coefficients of (2) are normalized, scaled, and converted to integers in RNS representation before computation of the distortion function, as discussed in section 2.6.

2.5 LIBRARY CONSTRUCTION

The change in method of spectral analysis from a full LPC-based analysis to a simpler analysis based upon the squared Euclidean distance of the autocorrelation coefficients simplifies the construction of the reference library in training mode. It is not necessary to compute and store the inverse correlation coefficients of the reference frames. The data now appended to the reference library for each utterance consist of the number of models (frames, segments) characterizing the utterance, a pointer to the utterance text string, and the models themselves. Each model (one per frame or utterance segment) is a data structure consisting of the P + 1 autocorrelation coefficients and a pointer to the next model. The coefficients are normalized, scaled, and represented by their residues modulo the n moduli of the chosen RNS before being stored in the library.

2.6 SQUARED EUCLIDEAN DISTANCE FUNCTION

The evaluation of a local distortion function to measure the degree of dissimilarity between a pair of reference and test frames is the most computationally intensive calculation performed in a DTW-based word recognition system. Hence it is the calculation that potentially benefits most from an RNS implementation. However, it is not sufficient that the computation consist mainly of additions and multiplications to be well-suited for RNS implementation. In an earlier phase of this study, the Itakura-Saito distortion metric was employed for computing local distances between reference and test frames. Even though the computation itself consisted chiefly of the calculation of a vector inner product, easily done in RNS, an unacceptably large range was required for the RNS to contain the calculation (chiefly the result of the large upscaling required to convert the inverse correlation coefficients to integer form). To prove practical and beneficial, an RNS implementation needs a distortion metric that can be computed in RNS without requiring a large dynamic range. In our recent work we successfully employed a squared Euclidean distance metric that directly uses the autocorrelation coefficients of the test and reference utterances.

The Euclidean distance computation consists of three steps:

1. Normalize the correlation coefficients

$$r_j' = r_j/r_0 \qquad j = 0, 1, \ldots, P$$

where P is the order of the autocorrelation model

2. Scale and convert to RNS representation

$$r_{ji} = \lceil S r_j' \rfloor \pmod{p_i} \qquad \left\{ \begin{array}{l} j = 0, 1, \ldots, P \\ i = 1, 2, \ldots, n \end{array} \right\}$$

where $\lceil x \rfloor$ denotes rounding to the nearest integer, S is the scale factor in use, and n is the number of moduli comprising the RNS

3. Compute the distortion value in RNS

$$d_i = \sum_{j=0}^{P} (r_{ji} - u_{ji})^2 \pmod{p_i} \qquad (i = 1, \ldots, n)$$

where d_i is the i-th component in the RNS representation of the distortion value d; r_{ji} and u_{ji} are the residues of the (normalized, scaled, and digitized) test and reference correlation coefficients, respectively.

Scale factors 8, 16, and 32 have been employed in simulations. Eight appears to be adequate for word recognition. With an input quantization of 12 bits and a scale factor of 8, an RNS range of 1000, or around 10 bits, is adequate to contain the distortion computation. This is a dramatic improvement over the 30 bits or more we found necessary to contain the Itakura-Saito metric calculation in our earlier work, and provides the key to a successful implementation of this calculation in RNS. Results concerning the determination of the RNS range are presented in expanded form in section 3.2.

2.7 QUANTIZATION OF DISTORTION VALUES BY PARTIAL MIXED-RADIX TRANSLATION

In [1], quantization of the distortion function values to a few levels was necessary to carry out the shortest-path computations without leaving RNS for magnitude comparisons. If the absolute differences of cumulative distances are less than half the largest modulus, then the magnitude comparisons can be made in the channel for the largest modulus and communicated to the remaining channels.

In [1], we presented a somewhat complex algorithm for performing the quantization by downscaling without leaving RNS. Now that the new distortion measure based on squared Euclidean distance is to be used, the range required for the distortion calculation is reduced to around 9 or 10 bits, permitting use of a modest sized three-modulus RNS. For such an RNS, an attractive alternative method for quantization is offered by partial mixed-radix translation of the distortion value residues.

The mixed-radix expression for an integer n represented in a three-modulus RNS composed of moduli p_1, p_2, and p_3, is

$$n = n_3 p_2 p_1 + n_2 p_1 + n_1 \quad (\text{mod } p_1 p_2 p_3). \tag{4}$$

If r_1, r_2, and r_3 are the residues of n mod p_1, p_2, and p_3 respectively, then

$$\begin{aligned} r_1 &= n \quad (\text{mod } p_1) = n_1 \\ r_2 &= n \quad (\text{mod } p_2) = (n_2 p_1 + n_1) \quad (\text{mod } p_2) \\ r_3 &= n \quad (\text{mod } p_3) = (n_3 p_2 p_1 + n_2 p_1 + n_1) \quad (\text{mod } p_3) \end{aligned} \tag{5}$$

so that

$$\begin{aligned} n_1 &= r_1 \\ n_2 &= (p_1^{-1}(r_2 - r_1)) \quad (\text{mod } p_2) \\ n_3 &= ((p_1 p_2)^{-1}(r_3 - n_2 p_1 - r_1)) \quad (\text{mod } p_3) \end{aligned} \tag{6}$$

gives the mixed-radix coefficients for *n* in terms of its residues.

This representation leads to a very natural definition of thresholds for quantization of the computed distortion values to two or three levels with a minimum of computation required to perform the quantization.

As shown in figure 2.6, two-level quantization is achieved by setting a threshold at p_1. Distortion values less than p_1 are mapped to zero, and all others are mapped to 1. Three-level quantization is achieved by setting two thresholds, one at p_1 and the other at $p_1 p_2$.

For two-level quantization, $d < p_1$ implies that $n_2 = n_3 = 0$. From (5), we see that $r_2 = r_1 \pmod{p_2}$ and $r_3 = r_1 \pmod{p_3}$. No calculation is required other than possible reductions of r_1 (mod p_2) and (mod p_3). The values of n_2 and n_3 need not be known; all we care about is whether they are zero or nonzero. The choice of p_1 as threshold may, of course, be inappropriate, but can be adjusted by the choice of the moduli set. Two-level quantization of the distortion values has proven to be adequate for word recognition, as shown by the simulation results in section 3.

IA-73.208

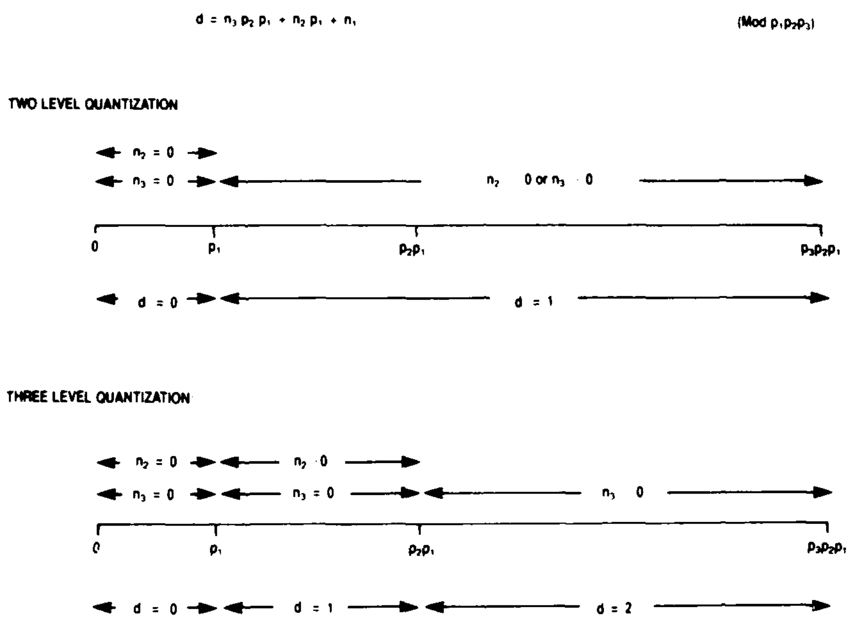


Figure 2.6. Quantization by Partial Mixed-Radix Conversion

Calculation of n_2 is required for three-level quantization by this scheme. Again, $d < p_1$ can be detected by testing whether $r_2 = r_1 \pmod{p_2}$ and $r_3 = r_1 \pmod{p_3}$. To detect when $p_1 \leq d < p_1p_2$, we have to distinguish the case ($n_2 = 0$ and $n_3 = 0$) from the case ($n_2 > 0$ and $n_3 = 0$). This requires calculation of n_2 by (6).

This calculation is simplified if the moduli are chosen so that p_1 (mod p_2) = $\pm$ 1, thus eliminating the multiplication by p_1^{-1}. The determination of whether n_3 is zero or nonzero can be simplified by choosing the moduli so that p_1 (mod p_3) = $\pm$ 1, eliminating the multiplication by p_1 in (6). The value of n_3 does not have to be calculated; we need only determine whether or not $r_1 + n_2 p_1 \equiv r_3$ (mod p_3). In our simulations, quantization to three levels has been less effective than quantization to two levels. The second threshold does not seem to be very helpful in our method of word recognition. Its principal effect is to increase the DTW scores of mismatches which already have high scores. On the other hand, the higher distortion values make a larger RNS necessary for the shortest path computations and/or lead to an increase in the frequency of overflow in these computations. These issues are discussed further in section 3.8.

2.8 SHORTEST-PATH COMPUTATIONS

A DTW algorithm finds the shortest path through a grid of points. Each point of the grid represents a matching of a selected pair of short-time segments, or frames, of the unknown test pattern and a given reference pattern. Associated with each grid point is a value that is the calculated local distortion for the particular match of test and reference frames represented by the point. Associated with each path through the grid is a distance that is a weighted sum of the local distortions for grid points lying on the path. The output of the DTW algorithm is a score, the length of the shortest path through the grid, representing the degree of dissimilarity between the matched patterns.

Three functions are required for dynamic time-warping: construction of the DTW grid, the set of points (i(k),j(k)) on which the DTW path is permitted to lie; evaluation of a local distortion measure for all points of the grid; and location of the shortest path through the DTW grid from the point (1,1) to the point (m,n), where m is the number of reference frames and n is the number of test frames to be matched. The shortest-path algorithm is a special case of dynamic programming [4]. In this section the determination of the DTW grid point set, given the number of reference frames m, the number of test frames n, and a set of local and global path constraints, is described, and the calculations required for finding the shortest path through the grid are derived for a particular choice of local constraints. The unknown test utterance is always assigned to the y-axis (vertical), and the reference utterance is assigned to the x-axis (horizontal).

2.8.1 DTW Path Constraints

Initially, before application of any constraints, the DTW grid (figure 2.7) consists of the $m \times n$ points (i,j), $1 \leq i \leq m$, $1 \leq j \leq n$. Each point (i,j) represents the matching of the i-th reference frame against the j-th test frame. Certain matches and sequences of matches, i.e. paths, may be unreasonable to make, however, and should be ruled out in advance. Rules are adopted in a speech recognition system to avoid such unreasonable paths and pointless computation. The local and global path constraints define these rules.

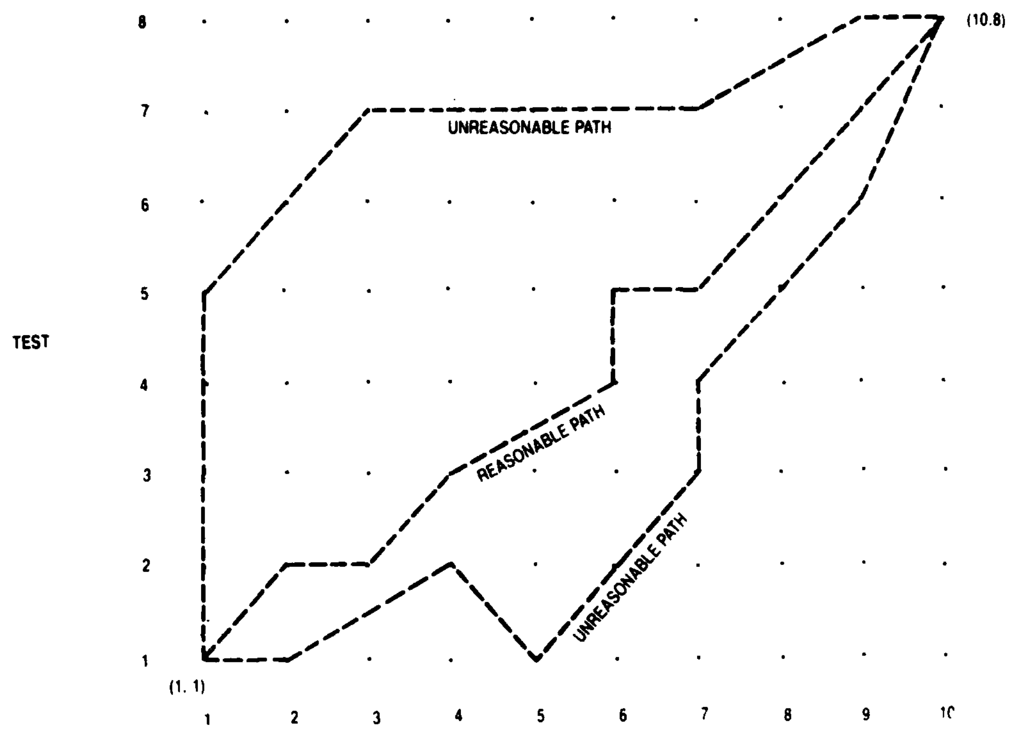


Figure 2.7. Unconstrained DTW Grid

Local path constraints specify the ways in which a particular path point (i(k),j(k)) can be reached from a preceding path point (i(k - 1), j(k - 1)). In accordance with [3], we represent allowed local paths by a set of productions from a regular grammar. A production is a rule of the form

$$P: (a_1,b_1)(a_2,b_2)\ldots(a_L,b_L) \tag{7}$$

where L is the length of the production, and the (a,b)'s are segments in a sequence of local moves. All a's and b's and L are assumed to be (small) nonnegative integers. Using a production, a local path to the point (i(k),j(k)) can be traced backwards to the point (i(k - 1), j(k - 1)) through L - 1 intermediate points:

k-th point: (i(k),j(k))

s-th intermediate point: $(i(k) - \sum_{\ell=1}^{s} a_\ell,\ j(k) - \sum_{\ell=1}^{s} b_\ell)$

(k - 1)st point: $(i(k - 1),j(k - 1)) = (i(k) - \sum_{\ell=1}^{L} a_\ell,\ j(k) - \sum_{\ell=1}^{L} b_\ell)$

This representation of local path constraints provides a great deal of flexibility in their choice. The left-hand side of figure 2.8 illustrates the type 3 constraints of [3], which are specified by the four productions

P1: (1,0)(1,1)
P2: (1,0)(1,2)
P3: (1,1)
P4: (1,2)

These four productions define four distinct possible local paths to a given point (i(k),j(k)) in the DTW grid, coming from the points (i(k) - 2, j(k) - 1), (i(k) - 2, j(k) - 2), (i(k) - 1, j(k) - 1), and (i(k) - 1, j(k) - 2), respectively. The first two of these local paths also pass through the intermediate point (i(k) - 1, j(k)). Note that for any local path to be valid, its starting point (i(k - 1), j(k - 1)) and its end point (i(k),j(k)) must belong to the valid point set.

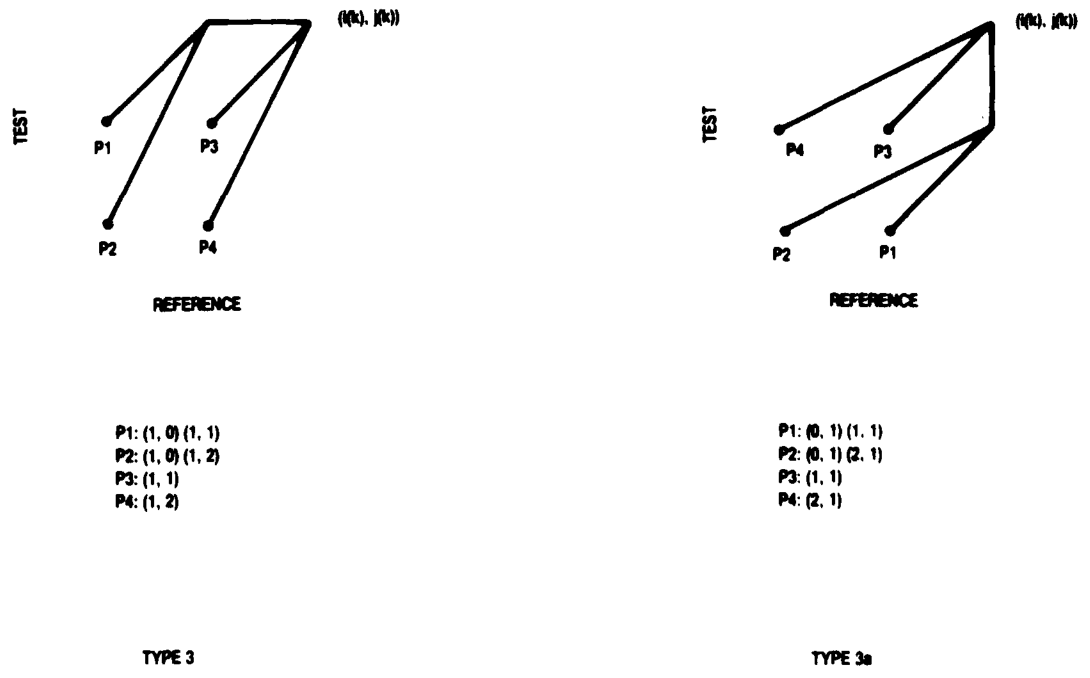


Figure 2.8. Local Path Constraints

A zero value for an a (b) in a production implies that the corresponding reference (test) frame is to be matched with more than one test (reference) frame. A value greater than one, on the other hand, results in one or more reference (test) frames being skipped (not matched) altogether. Thus, paths P1 and P2 of the type 3 constraints allow a given test frame to be matched with more than one reference frame, while paths P2 and P4 permit the skipping of a test frame. Under these constraints, each reference frame is used exactly once. Corresponding to the type 3 constraints is a reflected version, the type 3a constraints, shown in the right-half of figure 2.8. These are specified by the four productions

P1: (0,1)(1,1)
P2: (0,1)(2,1)
P3: (1,1)
P4: (2,1)

For these constraints, paths P1 and P2 match a given reference frame against more than one test frame, while paths P2 and P4 permit a reference frame to be skipped, but each test frame is used once and only once.

While there is no apparent reason for claiming that one set of constraints performs better than the other, it seems more natural to require that each test frame be matched exactly once, while allowing reference frames to be skipped or used more than once. Myers et al. in effect tested both types (along with a number of other sets of local constraints) by using the type 3 constraints but allowing the assignment of test and reference to the x- and y-axes to be reversed. They found better results for the reversed case, which corresponds to using the type 3a constraints. We used type 3a constraints in all simulations reported in section 3.

Associated with each local path to a grid point (i,j) is a path cost that is a weighted sum of the local distortion values for grid points passed through by the path. One of the simplest of weight functions takes the form

$$w(k) = i(k) - i(k - 1). \qquad (8)$$

For this weight function, used in all simulations reported in section 3, the weight assigned to a local path is the distance traversed in the reference direction, i.e., the sum of the a's in the production defining the local path. It is customary to divide the weight equally among the segments forming the path. Thus, for type 3 local constraints, this weight function assigns unit weights to all path segments, whereas for the type 3a constraints a fractional weight results for the segments of path P1.

Local constraints limit the valid point set making up the DTW grid in the following manner. For each procedure P of a local constraint, let sum(a) denote the sum of all the a's and let sum(b) denote the sum of all the b's. The slope of the local path is given by the ratio sum(b)/sum(a). Let e_{max} and e_{min} denote the maximum and minimum slopes, respectively, obtained over all productions comprising the local constraint. If we draw lines of slope e_{min} and e_{max} through the endpoints (1,1) and (m,n), the resulting four lines define a parallelogram in the initial DTW grid within which all valid points must lie (see figure 2.9). Points intermediate to local paths may lie outside this parallelogram, but the endpoints of such paths must themselves lie on or within the parallelogram. In figure 2.9 the parallelogram resulting from the type 3 constraints of [3] is shown, drawn in solid lines, representing ten reference frames and eight test frames.

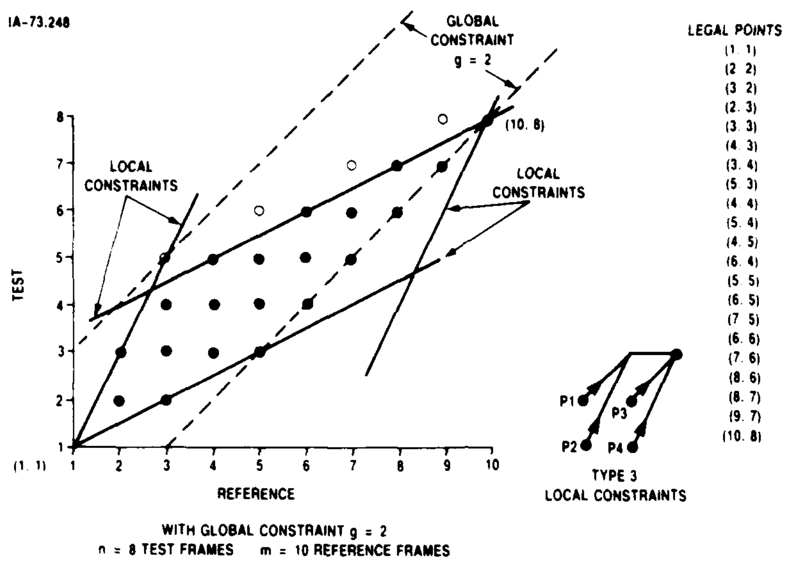


Figure 2.9. Grid for Type 3 Local Constraints

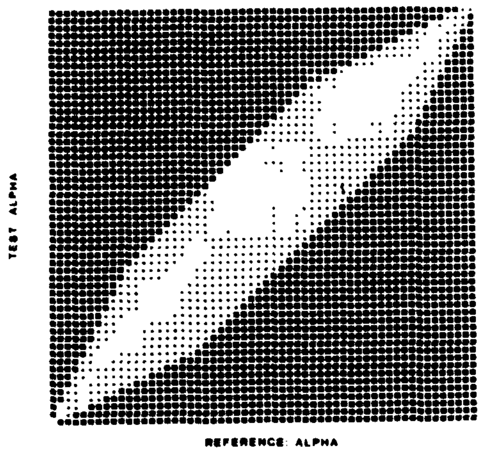
Global path constraints were introduced by Sakoe and Chiba [2] to further delimit the legal point set. These constraints take the form

$$|i(k) - j(k)| \leq g \qquad (9)$$

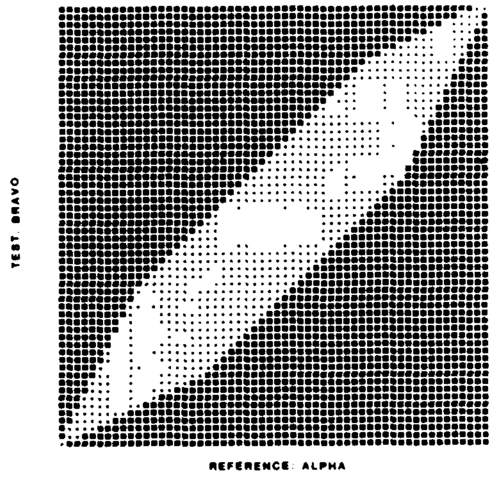
for some nonnegative integer g. They constrain the DTW path to lie within a corridor of width 2g centered on a 45° diagonal through the point (1,1). Of course, if $|m - n| > g$, then the endpoint (m,n) cannot satisfy the global constraint, and no legal DTW path can be found. In addition to restricting where the path can lie, the global constraint can be used to rule out altogether a search for the shortest path whenever the lengths of the test and reference utterances are too dissimilar.

A choice of $g = 0$ permits no path unless $n = m$, in which case all local paths must begin and end on the diagonal from (1,1) to (m,m). The global constraint usually limits the DTW grid by cutting off the interior corners of the parallelogram defined by the local constraints. In the example illustrated in figure 2.9, only the lower right corner is cut off by the severe global constraint $g = 2$. The resulting legal points comprising the DTW grid are shown as solid grid points. The hollow or empty points lying outside the parallelogram are intermediate points which may be passed through in traversing certain local paths that begin and end in the legal point set. The selected local distortion measure must be evaluated for such intermediate points as well as for the points in the legal set. A global constraint $g = 9$ has been employed for the simulations reported in section 3, except for section 3.7, where larger values have also been tried.

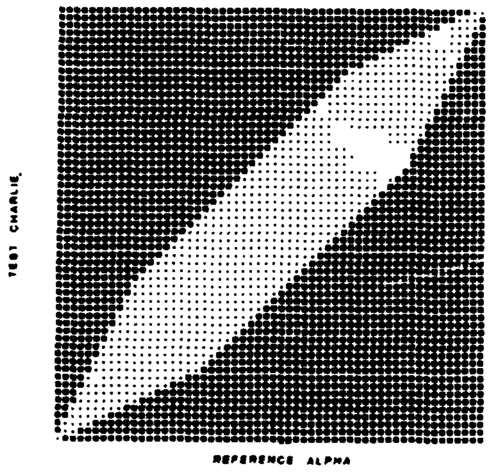
DTW grids from some simulations are shown in figure 2.10. For the four examples shown, type 3a local constraints were applied, with the global constraint set at $g = 9$. Single-threshold quantization of distortion values to 0 or 1 was applied. Points lying outside the legal point set and the allowed set of intermediate points



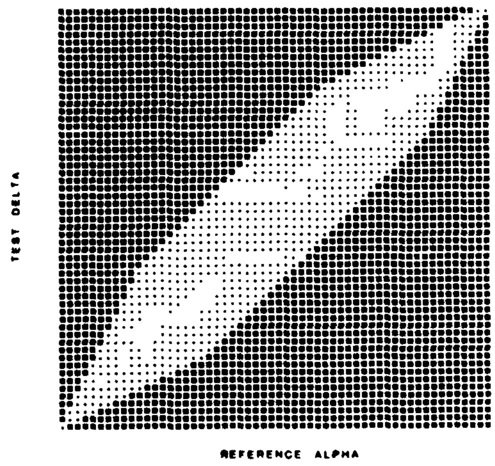
a. Alpha vs. Alpha



b. Bravo vs. Alpha



c. Charlie vs. Alpha



d. Delta vs. Alpha

Figure 2.10. Sample DTW Grids for Single-Bit Quantization of Distortion Values

are filled with a solid square. Points belonging either to the legal point set or the set of permitted intermediate points (bordering on the lower right-hand side of the legal set) are filled with a solid star or asterisk if the corresponding quantized distortion value is 1, and with a hollow star if the quantized distortion value is 0.

Figure 2.10a shows the DTW grid obtained for matching two similar words, a testset "alpha" against a library reference "alpha." A path with low cost can be found easily. The (normalized) DTW score for this match is .054545.

Figures 2.10b (test = "bravo") and 2.10d (test = "delta") show some similarity between the test pattern and the reference pattern for "alpha," but clearly no path can be found with so low a cost as in figure 2.10a. The resulting (normalized) DTW scores are .321429 for figure 2.10b ("bravo") and .327273 for figure 2.10d ("delta").

Figure 2.10c (test = "charlie") shows a very poor match between test and reference patterns. The resulting (normalized) DTW score is .821429.

2.8.2 DTW Path Computations

Dynamic time-warping for speech recognition was first formulated as a problem in dynamic programming by Sakoe and Chiba [2]. The problem of finding the best path through the DTW grid reduces to a special case of dynamic programming known as the shortest-route problem. This problem can be stated briefly as follows: Given a connected graph with two distinguished nodes A and B and with a cost associated with each arc from a node i to a node j of the graph, find the path, i.e., the sequence of arcs from A to B, whose summed cost is a minimum. Algorithms for finding an optimal solution to this problem were first given (independently) by Moore [5] and Dantzig [6]. Subsequently, Bellman [7] formulated the shortest-route problem as a dynamic programming problem.

The network, or graph, to which the shortest-route algorithm is applied is defined as follows: Nodes of the graph correspond to legal points of the DTW grid, with the grid point (1,1) as the node A and the grid point (m,n) as the node B. The arc costs are defined as weighted sums of local distortions obtained for matches of reference and test frames corresponding to grid points from node i to node j. For the type 3 local constraints and the weight function defined in equation (8), the costs defined for arcs of the network derived from the DTW grid have the form

$$\begin{aligned} c(P1) = c(P2) &= d_{i-1,j} + d_{ij} \\ c(P3) = c(P4) &= d_{ij} \end{aligned} \tag{10}$$

where d_{ij} is the local distortion calculated between the ith reference frame and the jth test frame.

The minimum cost, c_{ij}, for any path to the node (i,j) is computed (under type 3 constraints and weight function (8)) as

$$\begin{aligned} c_{ij} = \text{Min}\ \big(&d_{ij} + c_{i-1,j-1},\ d_{ij} + c_{i-1,j-2}, \\ &d_{ij} + d_{i-1,j} + c_{i-2,j-1},\ d_{ij} + d_{i-1,j} + c_{i-2,j-2}\big) \end{aligned} \tag{11}$$

where the first two terms of the minimization are the cost of reaching (i,j) by local paths P3 and P4, and the latter two terms are the cost of reaching (i,j) by local paths P1 and P2. Let

$$\hat{c}_{ij} = d_{ij} + \mathrm{Min}\left(c_{i-1,j-1},\ c_{i-1,j-2}\right). \tag{12}$$

Then

$$\hat{c}_{i-1,j} = d_{i-1,j} + \mathrm{Min}\left(c_{i-2,j-1},\ c_{i-2,j-2}\right) \tag{13}$$

and

$$c_{ij} = \mathrm{Min}\left(\hat{c}_{ij},\ d_{ij} + \hat{c}_{i-1,j}\right). \tag{14}$$

c_{mn}, the minimum cost for any path to node (m,n), is the score returned by the DTW algorithm.

The shortest-path computation for type 3 local constraints and weight function (8) can be summarized as follows:

1. Compute the local distortion d_{ij} from the test frame correlation coefficients $r_n(j)$ and the reference-frame correlation coefficients $u_n(i)$

2. Compute $\hat{c}_{ij} = d_{ij} + \mathrm{Min}\left(c_{i-1,j-1},\ c_{i-1,j-2}\right)$

3. Compute $c_{ij} = \text{Min} \left(\hat{c}_{ij}, d_{ij} + \hat{c}_{i-1,j} \right)$

We would like to employ RNS for step 1 because it is the most computationally intensive calculation in a DTW-based word recognition system. The problem that arises with RNS is the magnitude comparisons required for steps 2 and 3.

2.8.3 RNS Implementation of the Shortest-Path Computation

In order to make use of RNS for the local distortion calculations of a DTW algorithm, it is necessary to remain within RNS for the entire DTW shortest-path computation, leaving RNS only to convert the final score output by the algorithm for thresholding and comparison with other scores to select the best match. As discussed in section 2.8.2, solution of the shortest-path problem involves a sequence of additions and magnitude comparisons. In general, magnitude comparisons cannot be performed efficiently within RNS. However, the magnitudes being compared in the shortest-path computation may be similar. If their difference in absolute value does not exceed half the largest modulus in use, then relative magnitude can be determined without leaving RNS simply by testing the difference modulo this largest modulus.

In [1] we reported that these differences were not small enough to be contained within the range of a single modulus. However, further study showed that with suitable quantization of the local distortion values the differences could be kept within the range of a single modulus. In particular, quantization of the local distortion values to a single bit ("match" or "no-match") has proved very effective in our word recognition simulations. With quantization of distortion values, the revised shortest-path computation is as follows:

1. Compute d_{ij} from $r_n(j)$ and $u_n(i)$

2. Quantize to a single-bit d'_{ij} : 0 = match, 1 = no-match

3. Compute $\hat{c}_{ij} = d'_{ij} + \mathrm{Min}(c_{i-1,j-1}, c_{i-1,j-2})$

4. Compute $c_{ij} = \mathrm{Min}(\hat{c}_{ij}, d'_{ij} + \hat{c}_{i-1,j})$

The effects of overflows in the path computations are examined in section 3.8.

2.9 SELECTION OF THE WINNING TEXT

DTW scores calculated in RNS by the shortest-path algorithm are reconverted from RNS to conventional arithmetic representation and are normalized before deciding which is the winning text. Normalization is necessary to adjust for differences in utterance length; otherwise, reference texts with longer lengths tend to have larger DTW scores and are less likely to be accepted as candidates or winners than are shorter texts. The normalization factor used in all simulations reported in section 3 is min(m,n), where m is the number of reference frames and n is the number of test frames. Under the weight function (8) and employing single-bit quantization of distortion values, unnormalized DTW scores cannot exceed the number of reference frames m. Normalization will make most normalized DTW scores lie in the range 0 to 1, but when $n < m$, some normalized scores can be greater than 1.

After normalization of scores, the text with lowest score is designated as the winning text. In the error analysis performed subsequent to the simulations, an error is counted if this is not the correct text of the test or if two different reference texts are tied for lowest score.

SECTION 3

SIMULATION RESULTS

Results were obtained from many simulations performed to evaluate a DTW-based isolated word recognition system for which the essential time-warping signal processing functions are implemented in modular arithmetic. The database for building reference libraries and supplying test inputs to the simulation runs is described in section 3.1. Section 3 also describes the results of various simulations performed to evaluate various choices and parameter settings in the implementation design. The simulations discussed in section 3.9 employ a different database called the "rhymes" database, constructed for a study of word recognition when the vocabulary contains similar sounding words.

3.1 SIMULATION TEST SET

In [1] we used a speech database consisting of single-speaker utterances of eleven different words, the ten digits 0 - 9 plus "oh." We felt confident that the recognition results obtained from simulations using this database would be valid for larger databases, expecting that matches of similar words, i.e., different productions of the same word, would continue to exhibit low DTW scores relative to matches of different words. Nonetheless, it seemed desirable to repeat our simulation experiments on an expanded database in order to gain further confidence in the validity of this expectation. Therefore, the original test database was expanded by the addition of the 26 communication code words for the letters A - Z, giving us a vocabulary of 37 words.

Five different productions of the 37 words were recorded for use in the construction of reference libraries employed in the simulations. An additional sixth production of each word was recorded for the test input signals used in the simulations. Table 3.1 lists the 37 vocabulary words and shows the segment lengths of the six productions for each word of the vocabulary. In all simulation runs except those reported in section 3.7 the global constraint g was set at 9. No DTW score was computed if the number of reference and test frames differed by more than 9. This made some identifications harder to make than others. For example, "five" and "three" of the testset can be matched only with two correct members of the reference library, whereas many test inputs can be matched with five productions from the library. On the other hand, "foxtrot" cannot be incorrectly identified, for the global constraint rules out any matches between the test and an incorrect library utterance.

3.2 DETERMINATION OF THE RNS RANGE

A major breakthrough was achieved in FY85 in reducing the RNS range required for implementation of a DTW-based isolated word recognition system. Replacement of the Itakura-Saito distortion metric by a squared Euclidean distance computation employing a subset of the normalized autocorrelation coefficients of the test and reference segments reduced the RNS range required for the distortion computations from about 30 bits [1] to 9 or 10 bits, with no increase in recognition error rate and with improved discrimination between DTW scores for correct and incorrect matches.

Table 3.1

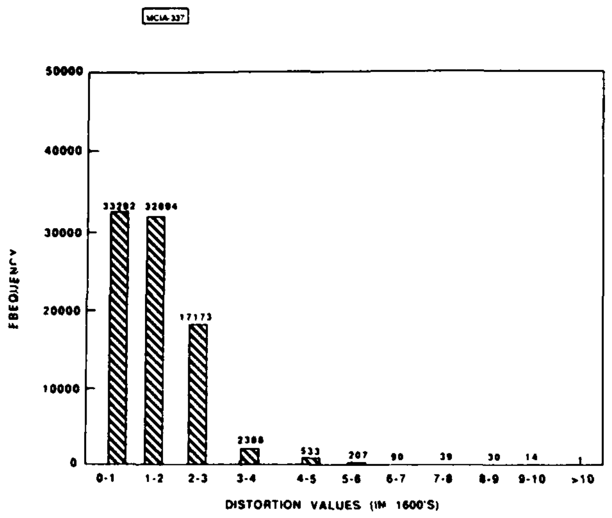
Number of Segments Produced for Five Reference Productions and One Testset Production by Utterance Database Segmentations

Word	Prodn1	Prodn2	Prodn3	Prodn4	Prodn5	Test
alpha	56	56	61	62	62	55
bravo	56	58	58	60	57	58
charlie	60	64	69	67	60	57
delta	58	51	53	56	54	55
echo	54	54	53	54	57	52
eight	56	53	50	54	47	50
five	78	67	82	82	82	72
four	59	60	58	58	61	55
foxtrot	91	103	101	103	104	99
golf	56	58	53	62	61	55
hotel	65	73	71	75	75	66
india	62	67	71	78	79	65
juliet	81	84	80	86	80	79
kilo	52	56	59	61	60	52
lima	55	58	62	62	58	58
mike	47	50	49	51	47	53
nine	62	60	57	56	61	60
november	75	77	76	78	75	73
oh	46	54	47	46	42	47
one	51	60	57	51	50	52
oscar	71	73	74	77	76	73
papa	57	64	78	71	64	57
quebec	58	62	66	69	64	67
romeo	77	70	71	70	68	69
seven	56	56	55	58	58	54
sierra	60	67	68	69	64	69
six	68	65	64	61	60	72
tango	64	69	67	66	68	66
three	57	57	66	64	61	50
two	54	48	49	51	46	50
uniform	77	84	78	82	82	80
victor	57	66	67	68	71	66
whiskey	59	60	68	67	64	60
xray	64	67	70	67	65	63
yankee	53	60	72	73	73	68
zero	53	59	57	67	61	57
zulu	61	63	61	64	63	58

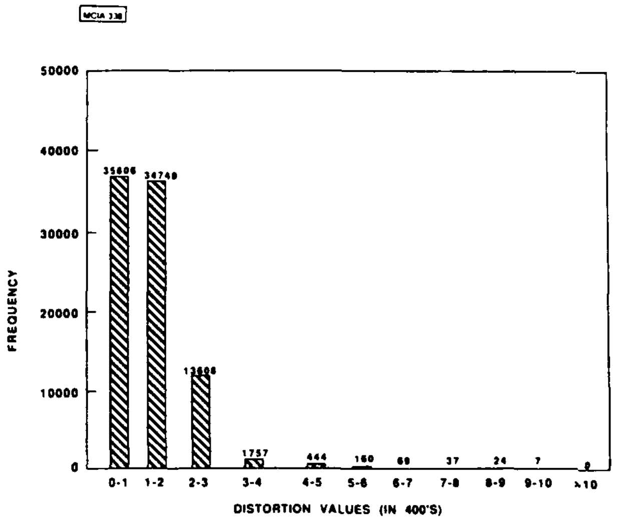
As discussed in section 2.6, the normalized correlation coefficients, lying between +1 and -1, are scaled before conversion to integers and RNS representation. Three scale factors were tried: 32, 16, and 8. For each scale factor, we computed histograms of the distortion values resulting from running the word "alpha" against the entire reference library. Plots of the resulting histograms are shown in figure 3.1. These plots show that RNS with respective ranges of 16000, 4000, and 1000 are adequate for containing the local-distortion computations when the normalized correlation coefficients are scaled by 32, 16, and 8, respectively. Since scaling by 8 has been shown by extensive simulation to be adequate for good discrimination between correct and incorrect word identifications, we have chosen 8 as our scale factor and used 8 in all simulations reported in the remainder of section 3. For these simulations we have employed two different 3-modulus RNS {23,11,2} with a range of 506, and {13,7,5} with a range of 455. Both have given excellent word recognition results even with occasional overflow of distortion values. We conclude that 9 bits are sufficient for performing the squared Euclidean distortion computation in RNS.

3.3 PRE-EMPHASIS OF SPEECH SAMPLES

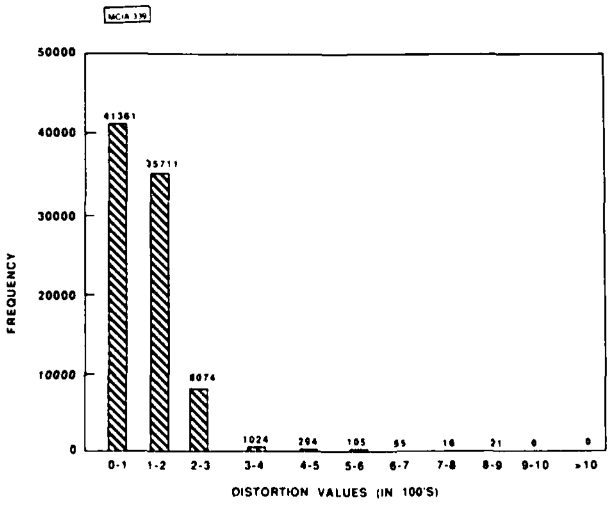
To study the effects of pre-emphasis of the speech samples we computed histograms of all normalized DTW scores produced by the shortest-path algorithm during a complete simulation run of the testset input against the stored reference library. The scores were divided into two classes depending on whether similar words (different productions of the same word) or different words (different text values) were being matched. Plots of the histograms



a. Scaling by 32



b. Scaling by 16



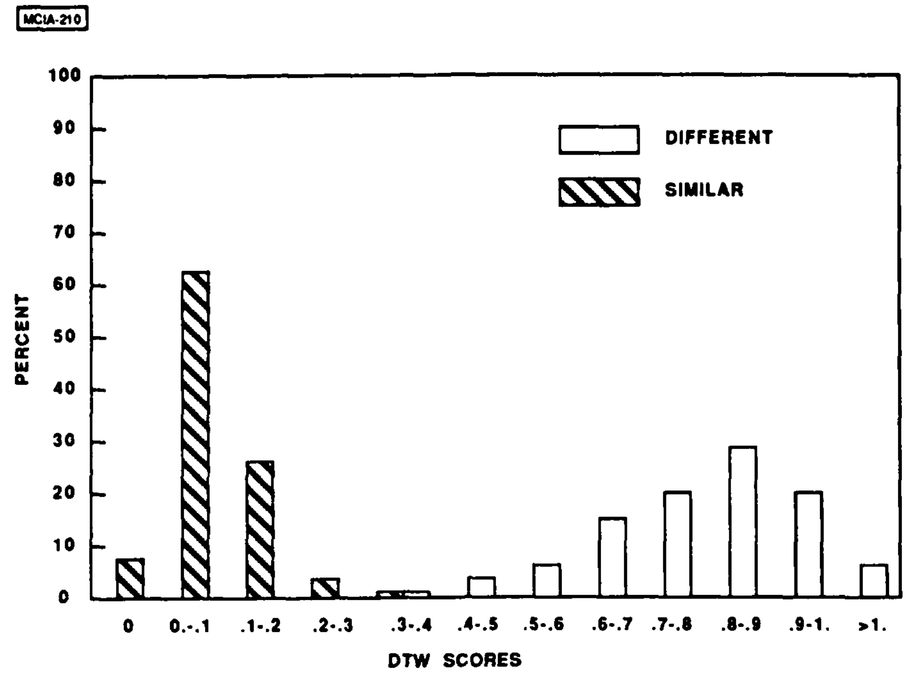
c. Scaling by 8

Figure 3.1. Histogram of Distortion Values

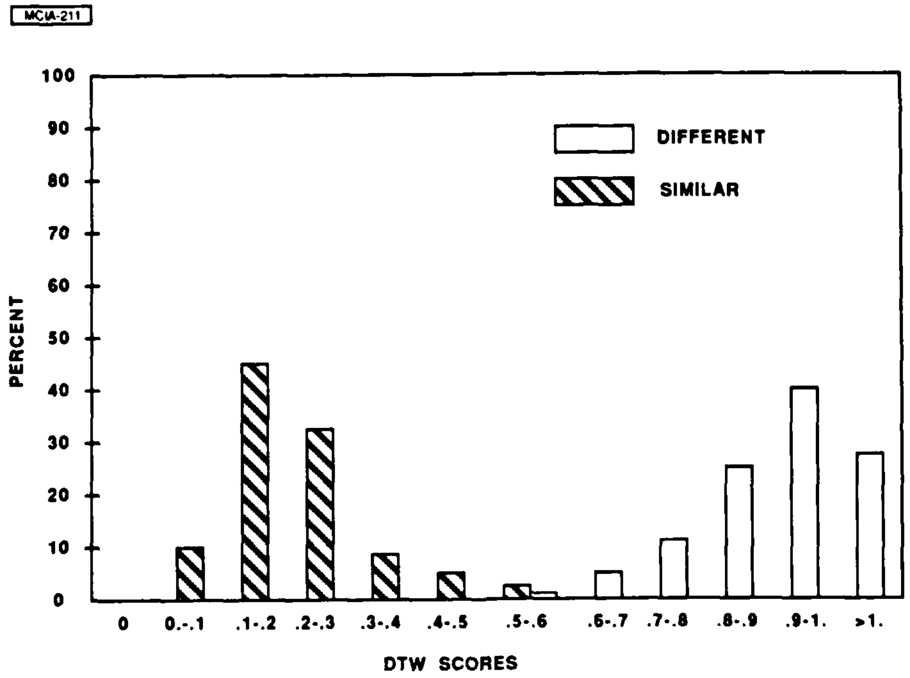
are shown in figures 3.2 (pre-emphasis applied to all speech samples, with $\mu = .95$) and 3.3 (no pre-emphasis applied). The a and b plots represent simulations for two distinct RNS: {23,11,2} and {13,7,5} respectively. (Single-bit quantization of distortion values was used for all simulations. PORDER is the order P of the autocorrelation model). The histogram heights represent the percentage of all matches of similar words (different words) whose normalized DTW scores fall into the respective classes 0, 0 to .1, .1 to .2, ..., .9 to 1., and greater than 1. (Normalized DTW scores > 1 result sometimes when the number of reference frames exceeds the number of test frames. The normalization factor employed is min(m,n).) Percentages have been rounded to the nearest integer value before plotting, so that percentages less than .5 do not show.

Fiqure 3.2 illustrates why the DTW algorithm is working so well in this implementation: there is excellent separation between the scores for similar matches (correct identifications) and those for different matches (incorrect identifications). In general, there are several correct matches in the library for each test input, and the best match is selected as the winning candidate. Decisions, therefore, tend to be made at the left or low-end of the histogram of similar words where, in this case, there is no overlap with the histogram of different words.

Comparison of figure 3.2 with figure 3.3 shows that pre-emphasis of the speech samples is helpful for better separation between the DTW scores for matches of similar words and those for matches of different words. For the simulation shown in figure 3.3b, a single recognition error oc-urred (no errors for the other three cases);

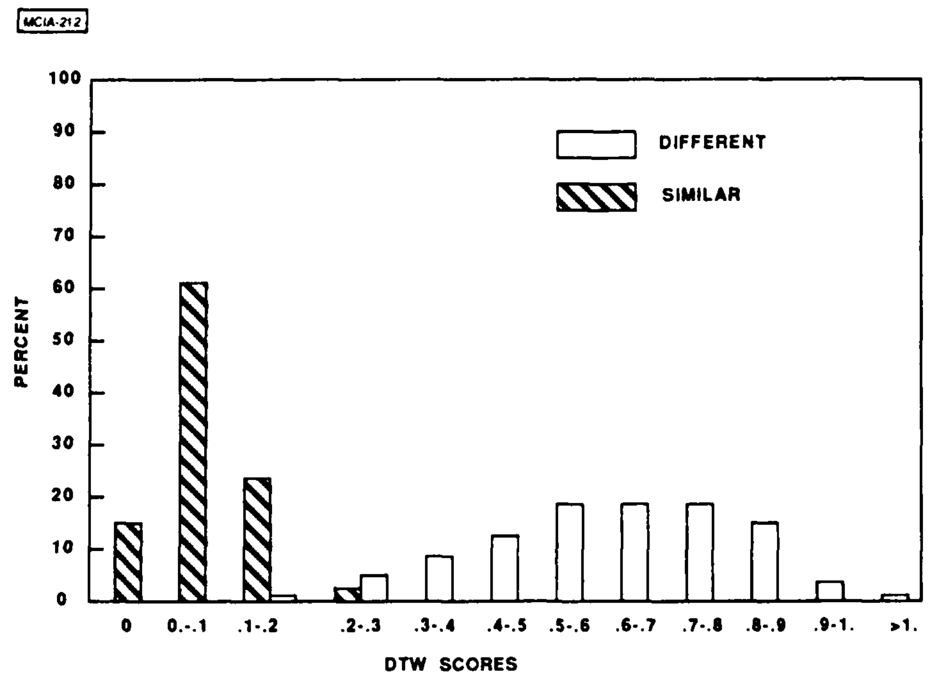


a. RNS {23,11,2}

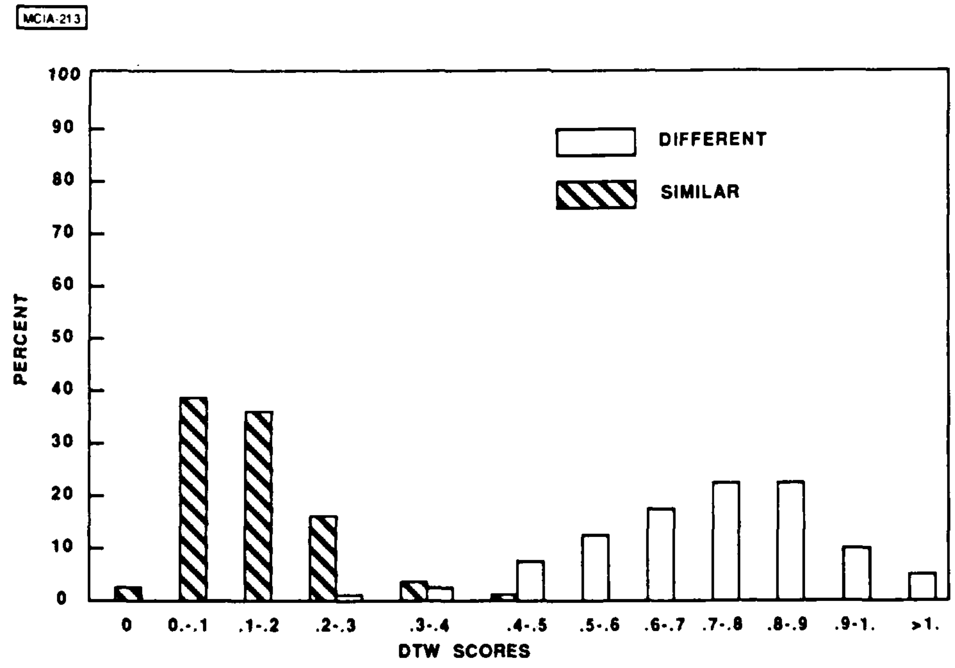


b. RNS {13,7,5}

Figure 3.2. DTW Scores with Pre-Emphasis: μ = .95, 10 Bit Samples, PORDER = 12



a. RNS {23,11,2}



b. RNS {13,7,5}

Figure 3.3. DTW Scores without Pre-Emphasis
10 Bit Samples, PORDER = 12

for the simulation shown in figure 3.2a, the smallest ratio of lowest wrong score to lowest correct score exceeded 3.7. (This ratio is ≤ 1 if an error occurs.)

The difference between the a and b plots reflects the choice of threshold. For the a plots, the quantization threshold is set higher (relative to the RNS range) than for the b plots. This results in lower DTW scores, both for similar and for different matches, shifting both histograms to the left and, in the case of 10-bit input quantization, providing better separation. With a coarser input quantization (< 10 bits), the lower setting of the threshold for the b plots begins to give increasingly better performance relative to the a plots.

3.4 INPUT QUANTIZATION

Although the A/D converters in our input preprocessing system yield 16-bit speech samples, it has long been apparent to us that 16 bits are not necessary for acceptable word recognition. Among the advantages of shorter samples is a reduction in the range needed for autocorrelation analysis in RNS. However, since the autocorrelation vectors are then normalized and scaled before the local distortion computation, the range needed for the latter is not affected.

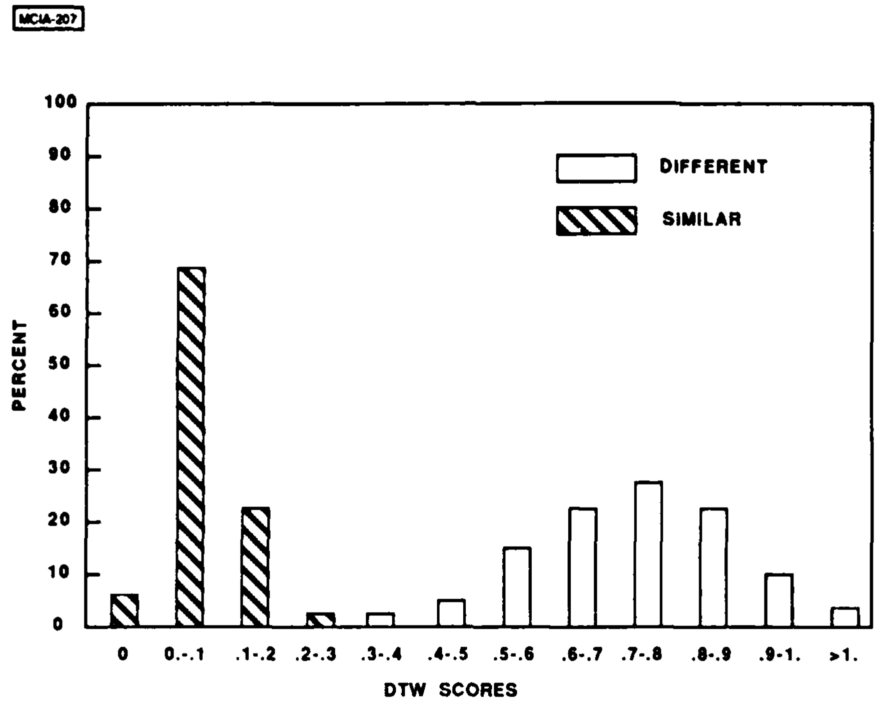
We have looked at reducing the input quantization from 10 bits to 8, 6, 4, 3, and 2, and also at increasing it to 12 and 16. We expected that a coarser input quantization, i.e., fewer sample bits, would lead to lower distortion values, hence to lower DTW scores and a left-shift of the score histograms. This is shown in the results of the simulations. Pre-emphasis was employed in all simulations. We report on a subset of the results.

Plots for 8-bit input quantization are shown in figures 3.4a (RNS {23,11,2}) and 3.4b (RNS {13,7,5}). Comparison with figures 3.2a and 3.2b for 10-bit quantization shows a slight shift of the histograms to the left for the coarser input quantization, but no apparent loss in discriminability between correct and incorrect matches.

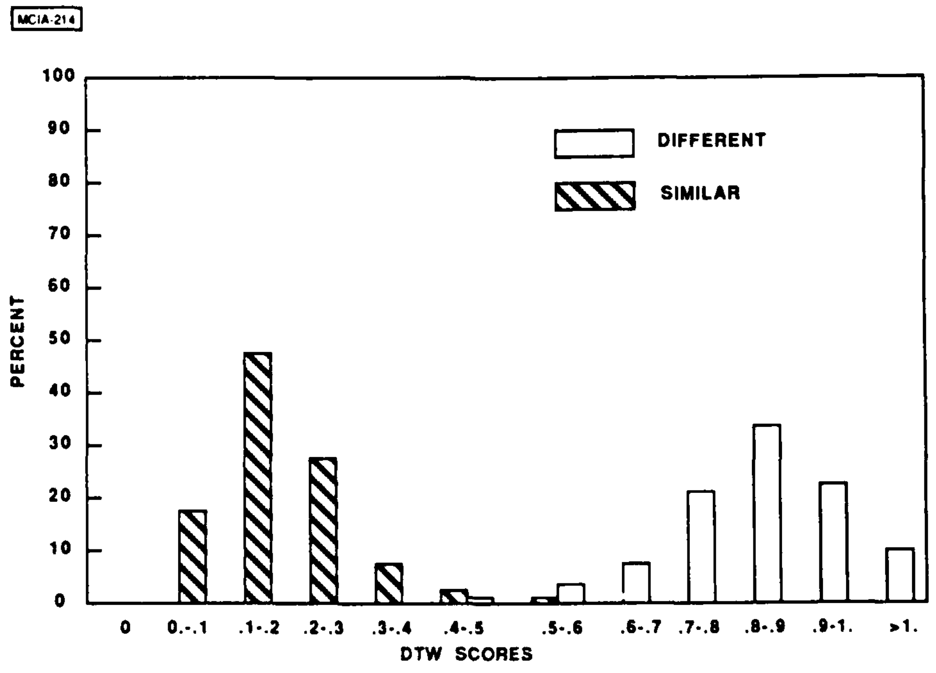
The picture changes somewhat when the quantization is reduced to 6 bits. Again, the reduction in sample size results in a left-shift of the histograms. For the RNS {13,7,3} (figure 3.5b) the separation between scores for matches of similar words and matches of different words remains good, but for the other RNS {23,11,2} the natural threshold setting $T = p_1 = 23$ is now too high relative to the reduced local distortion values and produces a considerable overlap in the resulting DTW scores. Both simulations resulted in a single recognition error, failing to distinguish "three" from "two" (RNS {23,11,2}; test = "three"), and "lima" from "tango" (RNS {13,7,5}; test = "lima"). In both cases, tie scores resulted.

Figure 3.6 shows histograms of DTW scores for 4-bit input quantization. The overlaps of the histograms for different and similar matches are now considerable, especially for the RNS {23,11,2}. This overlap leads to confusion in the word recognition process.

Figure 3.7 is a plot of the number of recognition errors occurring versus the input quantization for all simulation runs. Two curves are drawn, one for the RNS {23,11,2} and one for the RNS 13, 7, 5. A recognition error is counted in the error analysis of a simulation run whenever the selected text for identification of the test input is incorrect or whenever there is a tie in DTW scores between the best guess and the next best guess.



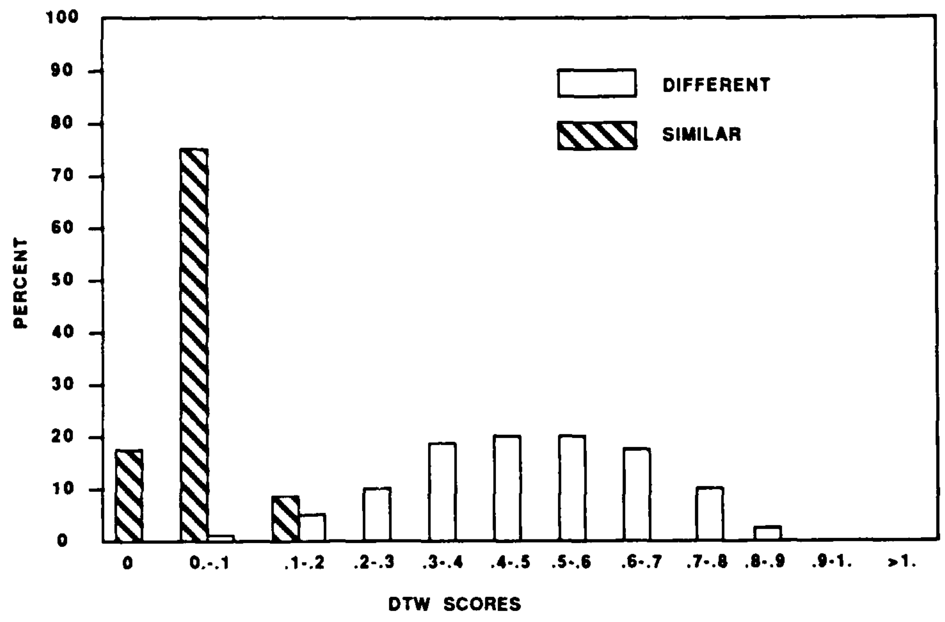
a. RNS {23,11,2}



b. RNS {13,7,5}

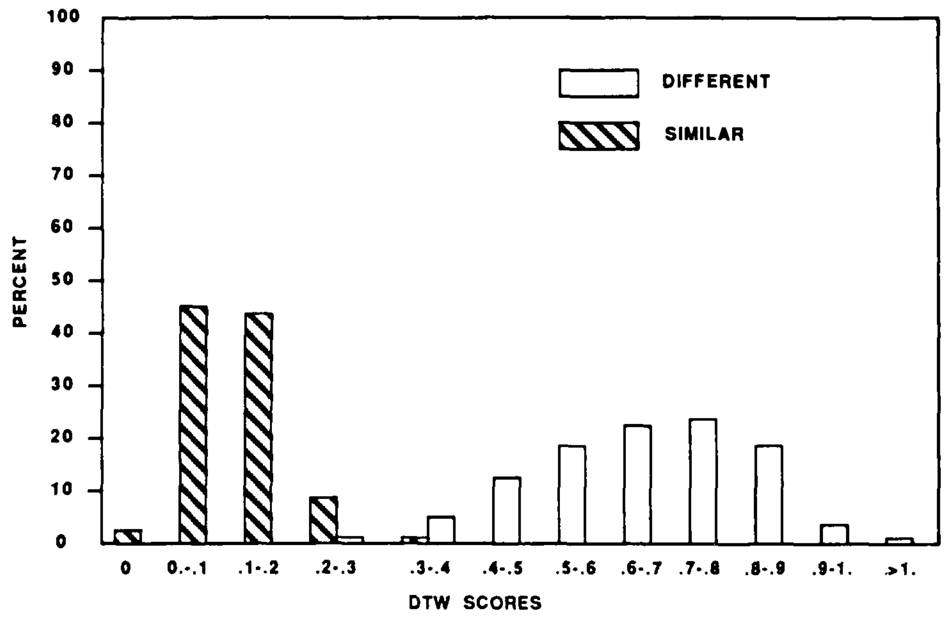
Figure 3.4. DTW Scores with Pre-Emphasis: μ = .95, 8 Bit Samples, PORDER = 12

MCIA-215



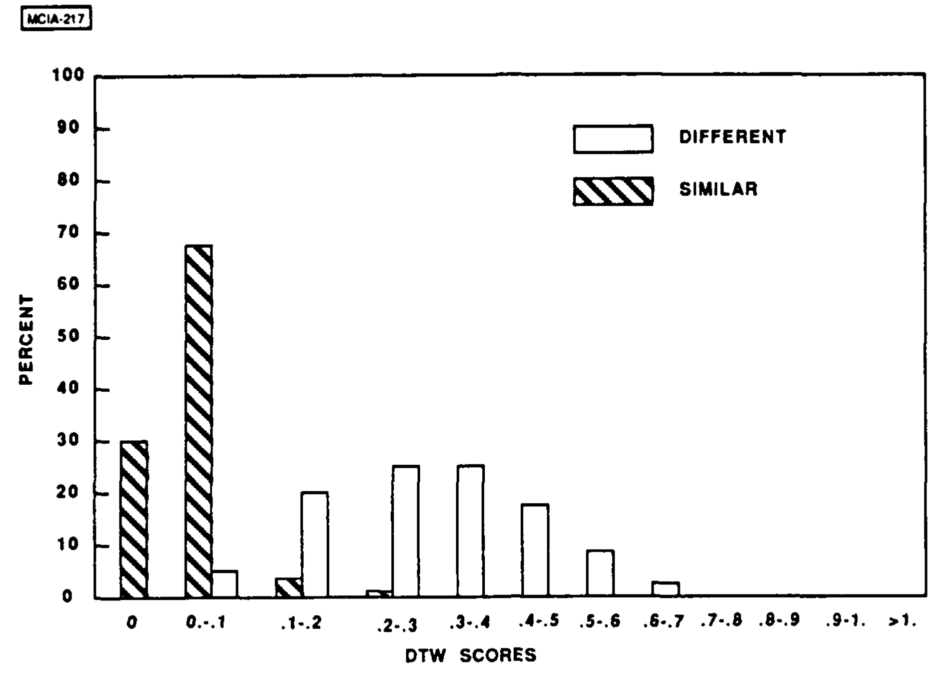
a. RNS {23,11,2}

MCIA-216

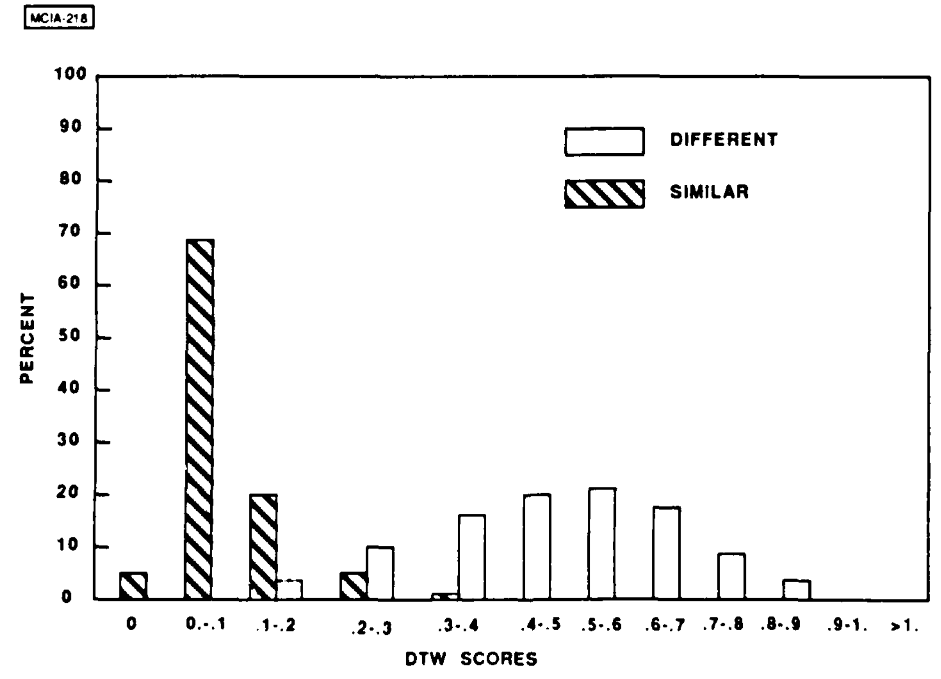


b. RNS {13,7,5}

Figure 3.5. DTW Scores with Pre-Emphasis: μ = .95, 6 Bit Samples, PORDER = 12



a. RNS {23,11,2}



b. RNS {13,7,5}

Figure 3.6. DTW Scores with Pre-Emphasis: u = .95, 4 Bit Samples, PORDER = 12

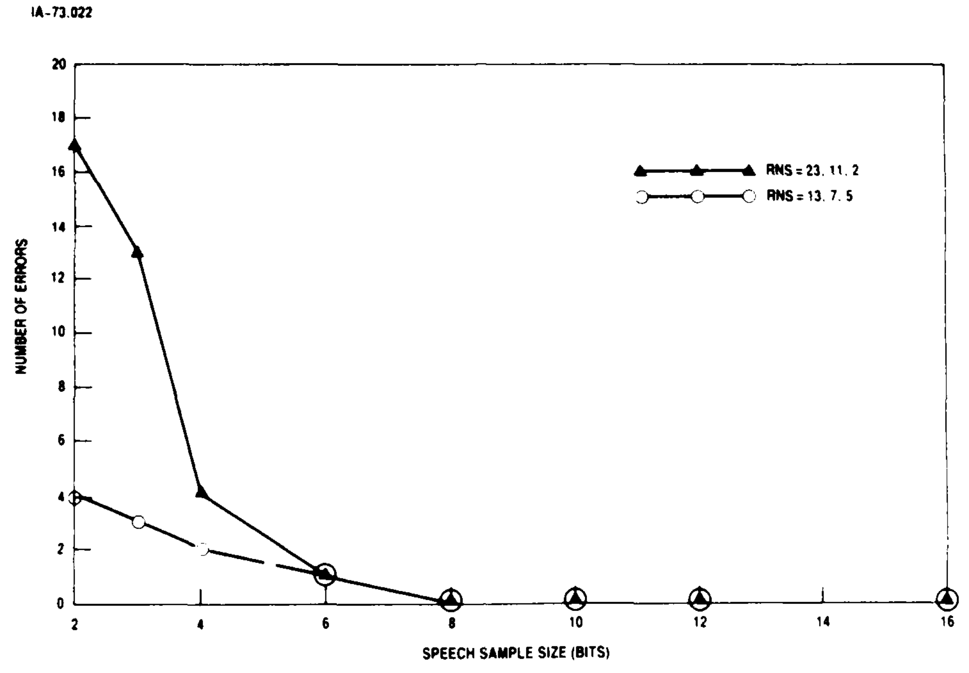


Figure 3.7. Error Analysis of Input Quantizations

We have used two other types of recognition criteria for comparing different simulation results. The first is called the minimum discrimination ratio, r_m. For each test utterance j we define a test discrimination ratio, r_j, by

$$r_j = \left\{ \begin{array}{ll} 0 & \text{if } w_j = 0 \text{ or } w_j < c_j \\ w_j/c_j & \text{otherwise} \end{array} \right\} \tag{15}$$

where c_j is the best DTW score for a reference with correct text and w_j is the best DTW score for a reference with incorrect text. The minimum discrimination ratio is defined by

$$r_m = \underset{j}{\text{Min}}\ r_j . \tag{16}$$

A plot of r_m versus input quantization is given in figure 3.8. Two curves are drawn, one for the RNS {23,11,2}, and one for the RNS {13,7,5}, showing the former gives better worst-case discrimination if the sample size is adequate (8 bits or more). No meaningful comparisons can be made when the quantization is 6 bits or less, so the curves are plotted only for sample sizes greater than 6.

IA-73.051

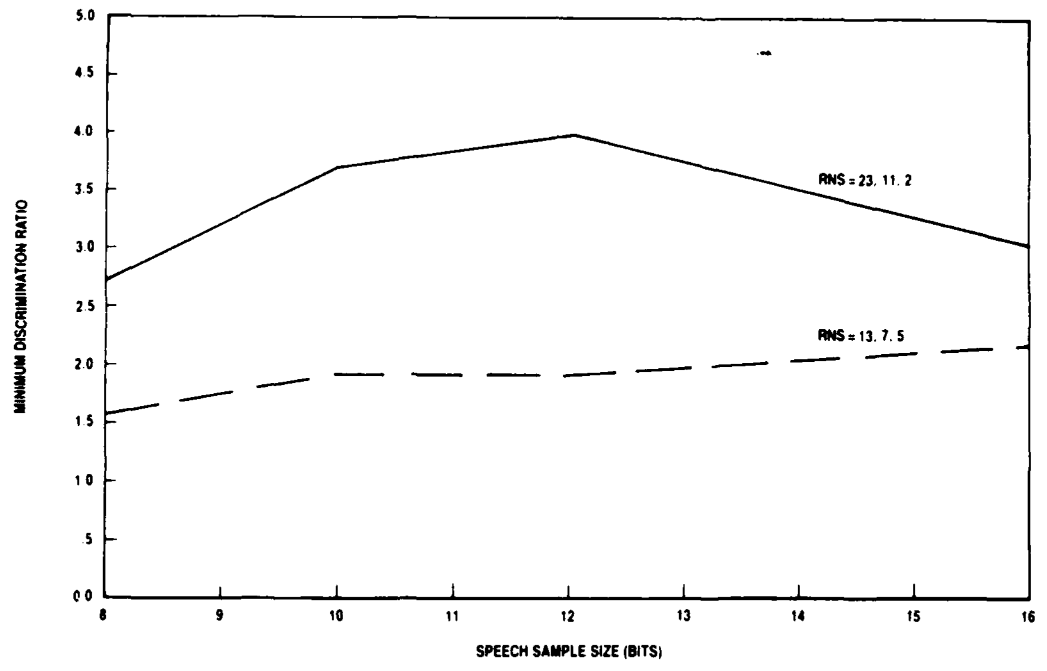


Figure 3.8. Minimum Discrimination Ratios for Input Quantizations

The third criterion used is an ensemble discrimination ratio r_e. Since some c_j will be zero, we cannot define an average discrimination ratio. The ensemble discrimination ratio is defined by

$$r_e = (\sum_j w_j)/(\sum_j c_j) \tag{17}$$

ignoring cases where no score is computed for a second-best text, e.g., "foxtrot". Again, this statistic is not meaningful for sample sizes less than 8 and has been plotted in figure 3.9 only for input quantizations 8 to 16.

To summarize these results, the DTW score histograms show excellent separation between scores for matches of similar words and those for matches of different words so long as the input quantization is adequate (8 bits or more). For coarser quantizations of the speech samples the two histograms exhibit more serious overlaps, giving rise to recognition errors (exceeding 10% for 2-bit quantization and the RNS {13,7,5}).

3.5 ORDER OF THE AUTOCORRELATION MODEL

The selection of the order of the autocorrelation model is important. It should be set as small as is consistent with good recognition performance, for both the size of the reference library and the time required for performing the local distortion calculation are approximately linear functions of the order P. In this section, we report on the results of simulations carried out to investigate the effects of changing P, i.e., PORDER. In general, as

IA-73.050

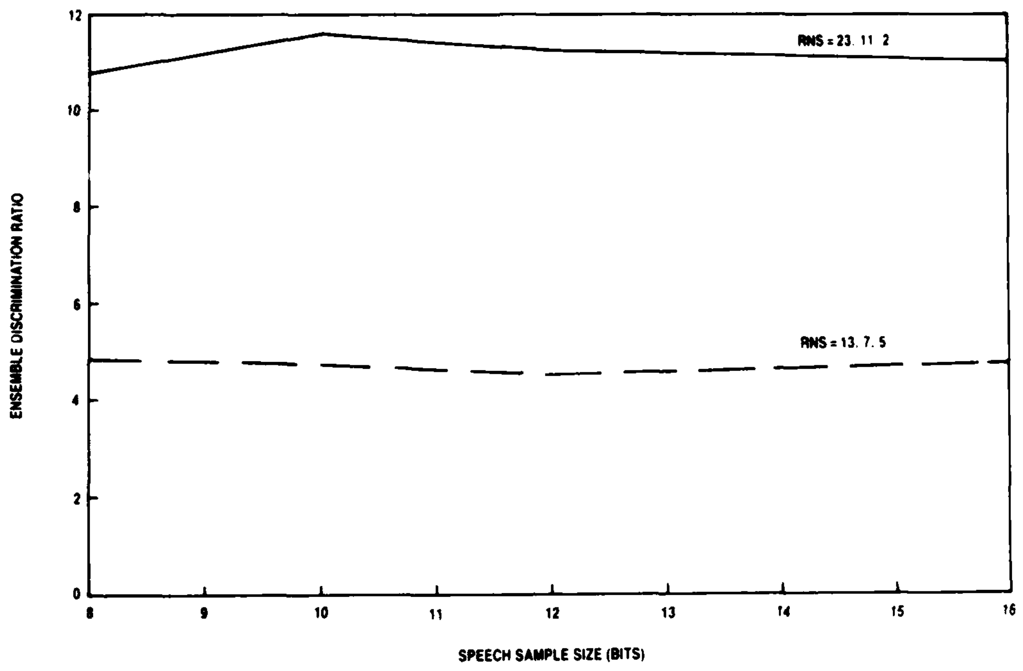


Figure 3.9. Ensemble Discrimination Ratios for Input Quantizations

PORDER is decreased, local distortion values decrease and, hence, the DTW scores are lower, shifting the histograms to the left.

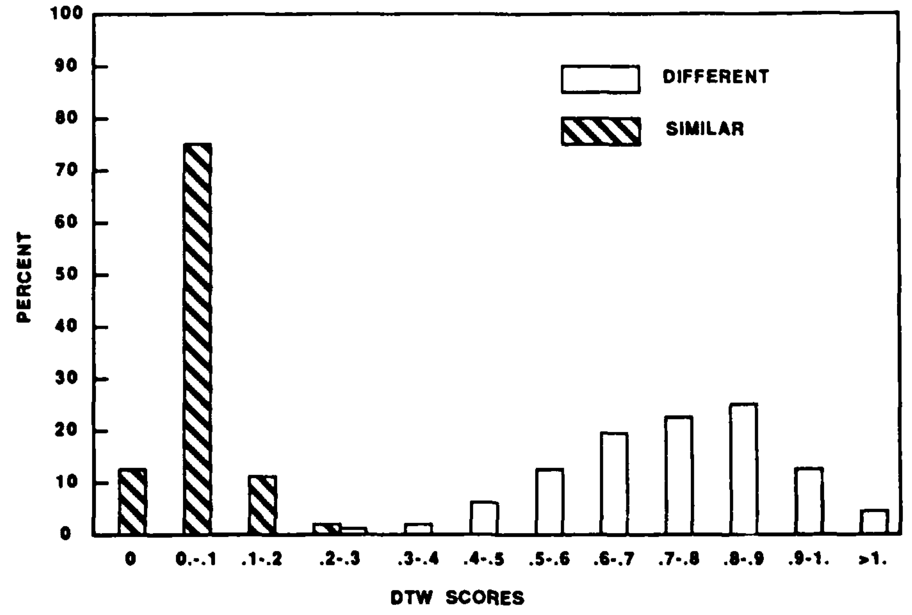
Figure 3.2 showed the DTW score histograms produced when PORDER = 12. Figures 3.10 through 3.13 show the effects upon the histograms where PORDER changes from 12 to 10, 8, 6, and 4, respectively. Figures 3.14 and 3.15 show the effects of increasing PORDER to 16 and 20, respectively. For the RNS {13,7,5}, separation between histograms for matches of similar words and matches of different words is best for PORDER = 12, although separation deteriorates only slightly for PORDER = 10 and 8. For the RNS {23,11,2} separation is best for PORDER = 12 and PORDER = 20, but again the overlaps for PORDER = 10 and 8 are small, and these appear to be acceptable choices.

In the error analysis of the simulation runs, recognition errors occurred only when PORDER = 4 for the RNS {23,11,2}. For this single case, three recognition errors were counted.

Figure 3.16 is a plot of r_m versus PORDER. Two curves are shown, for the RNS {23,11,2}, and the RNS {13,7,5}. This indicator of worst-case performance is optimized for the choices PORDER = 12 and RNS {13,11,2}. For the other RNS {13,7,5}, r_m is largest for PORDER = 8.

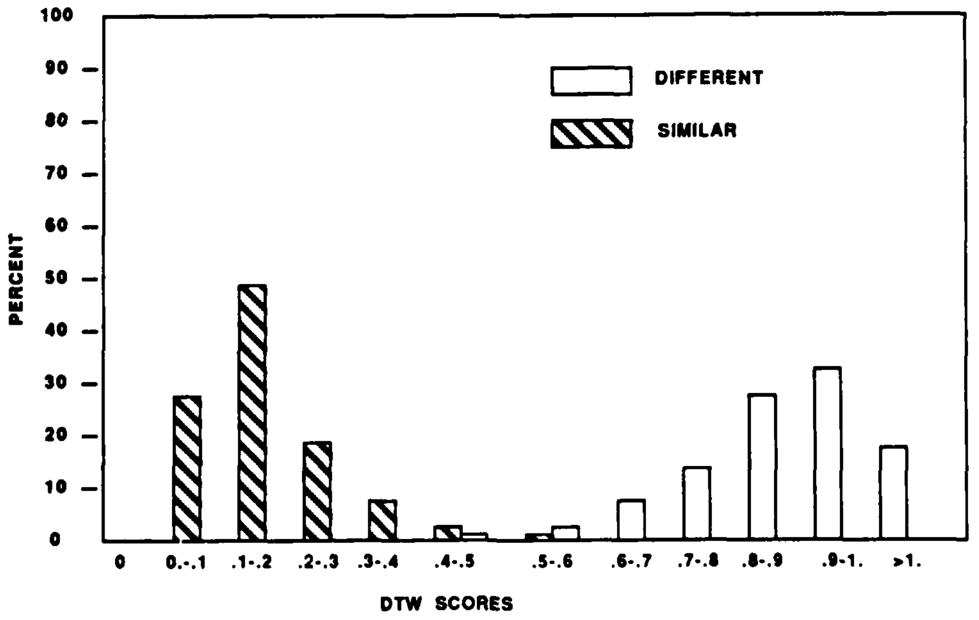
Figure 3.17 shows plots of r_e versus PORDER for each of the two RNS employed.

MCIA-195



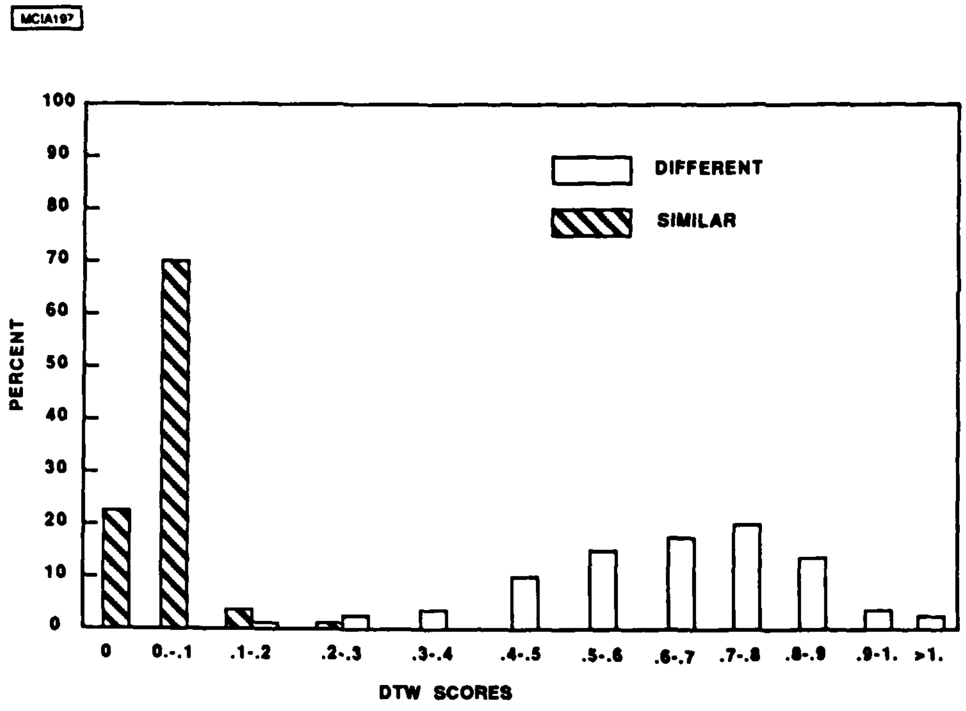
a. RNS {23,11,2}

MCIA-196

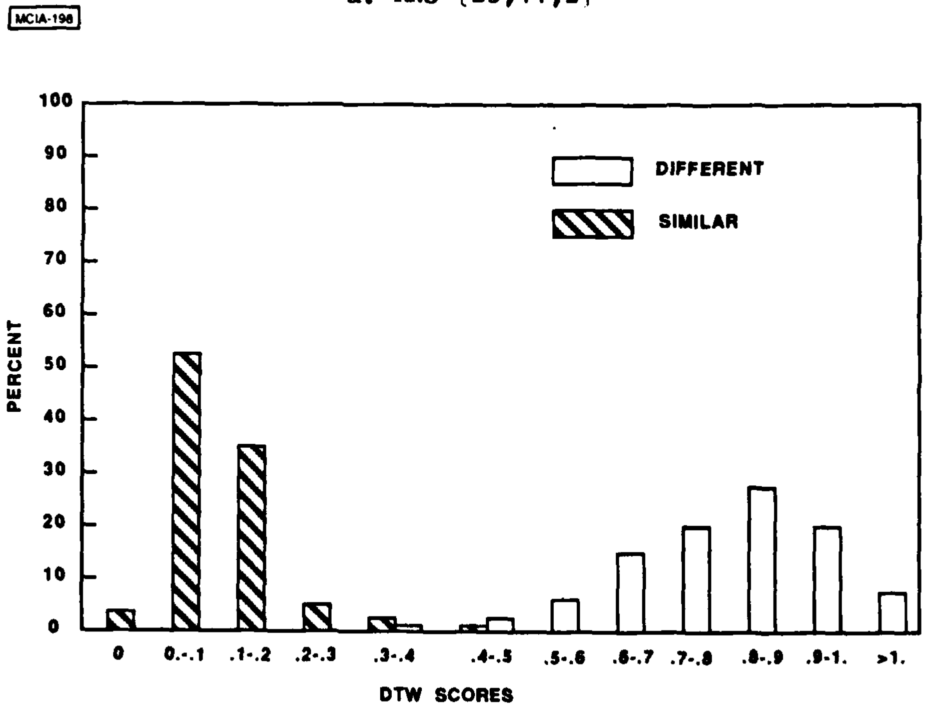


b. RNS {13,7,5}

Figure 3.10. DTW Scores with Pre-Emphasis: μ = .95, 10 Bit Samples, PORDER = 10



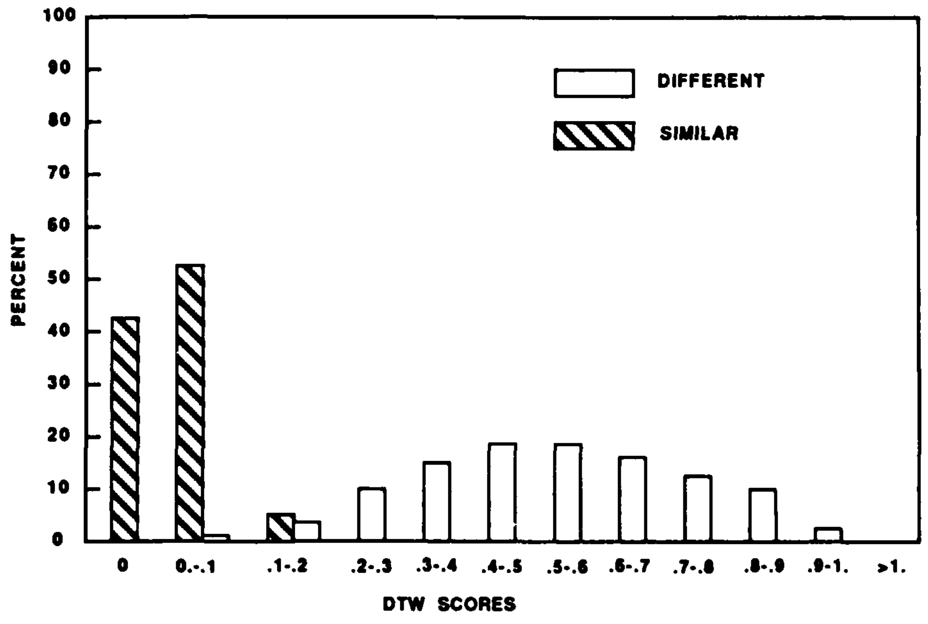
a. RNS {23,11,2}



b. RNS {13,7,5}

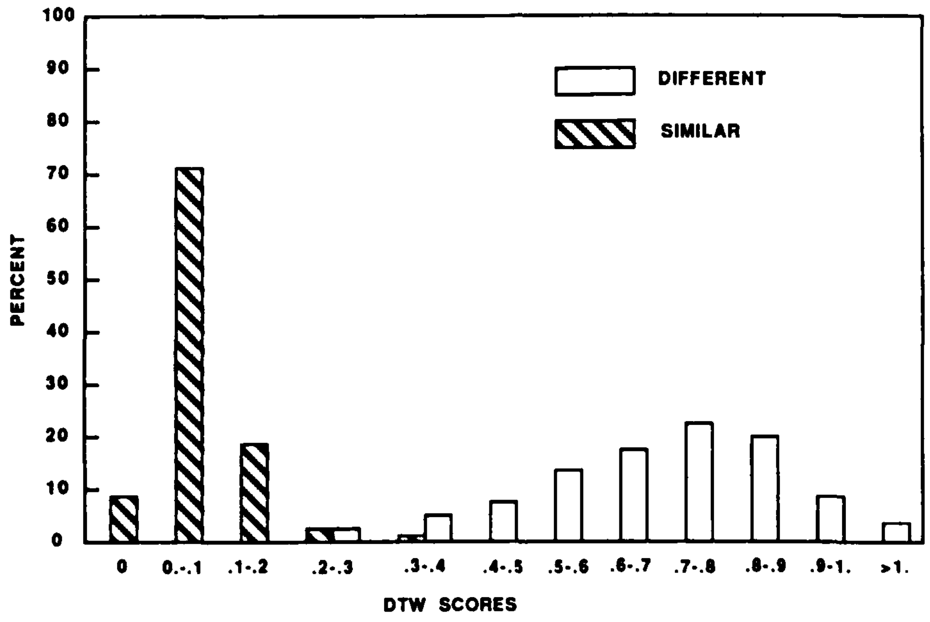
Figure 3.11. DTW Scores with Pre-Emphasis: μ = .95, 10 Bit Samples, PORDER = 8

MCIA-199



a. RNS {23,11,2}

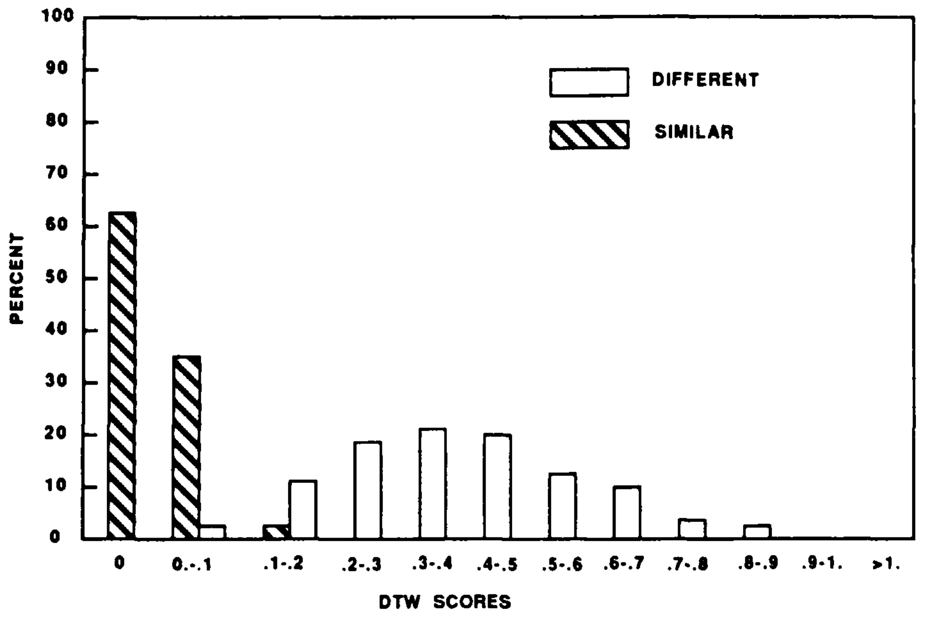
MCIA-200



b. RNS {13,7,5}

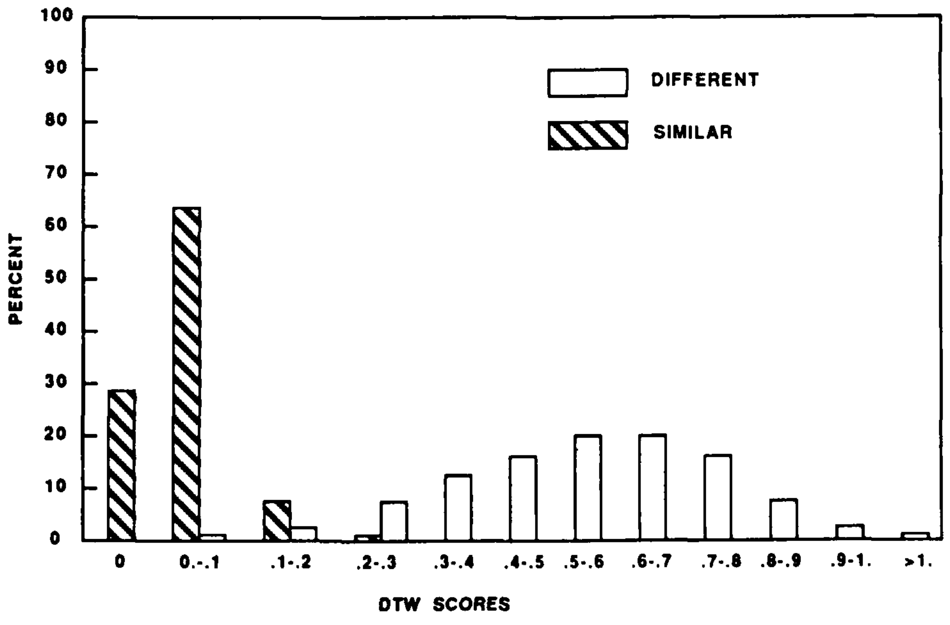
Figure 3.12. DTW Scores with Pre-Emphasis: μ = .95, 10 Bit Samples, PORDER = 6

MCIA-201



a. RNS {23,11,2}

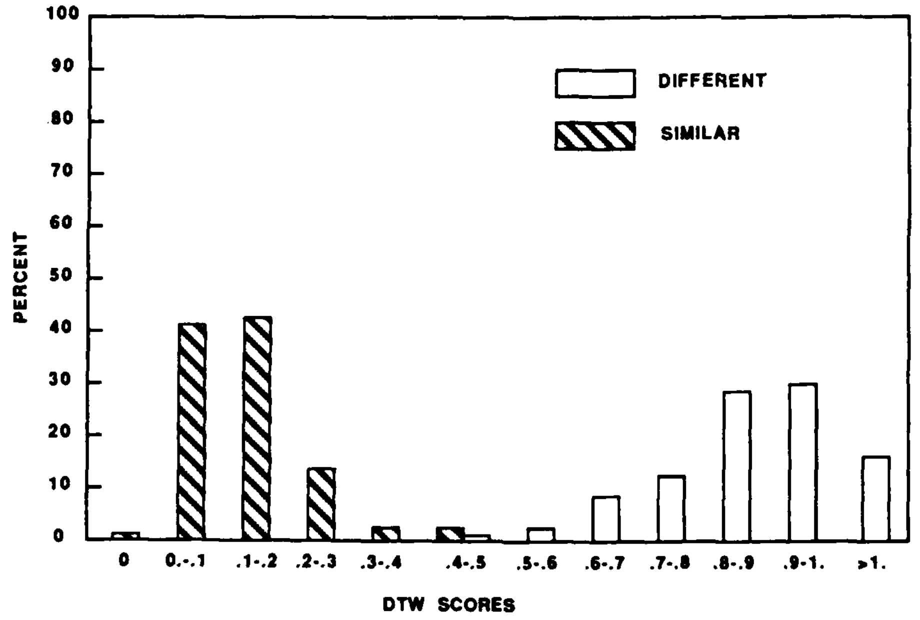
MCIA-202



b. RNS {13,7,5}

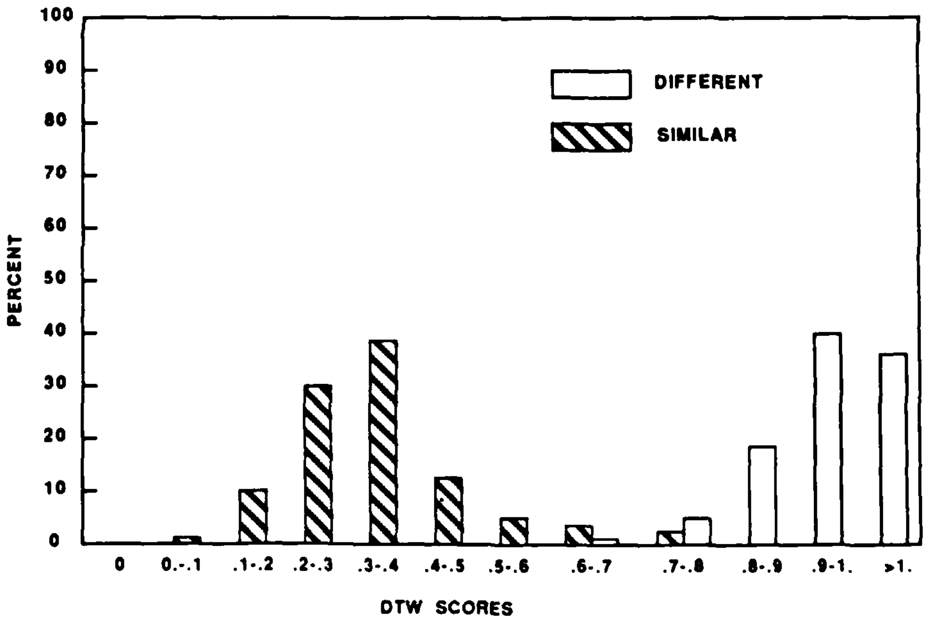
Figure 3.13. DTW Scores with Pre-Emphasis: $\mu = .95$, 10 Bit Samples, PORDER = 4

MCIA-203



a. RNS {23,11,2}

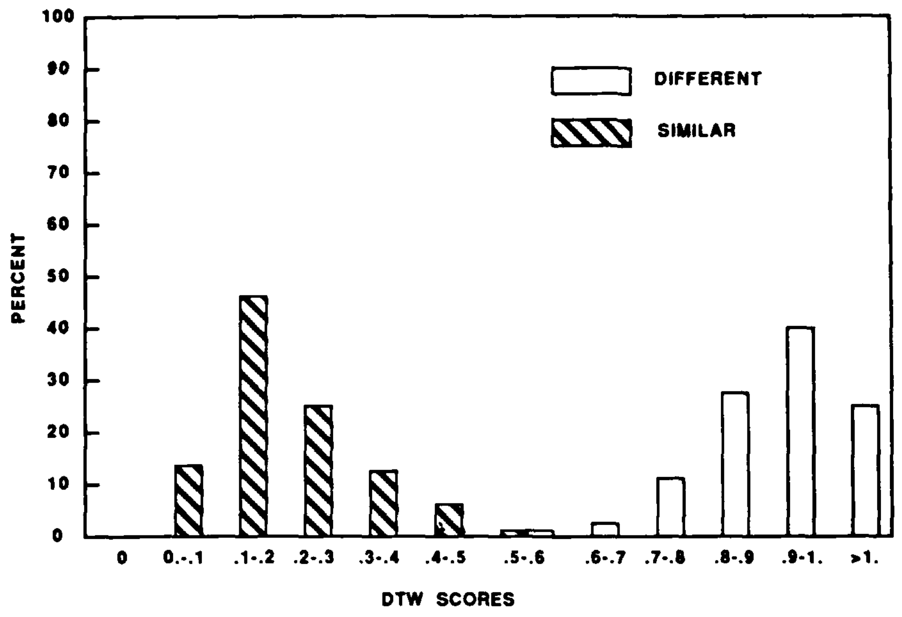
MCIA-204



b. RNS {13,7,5}

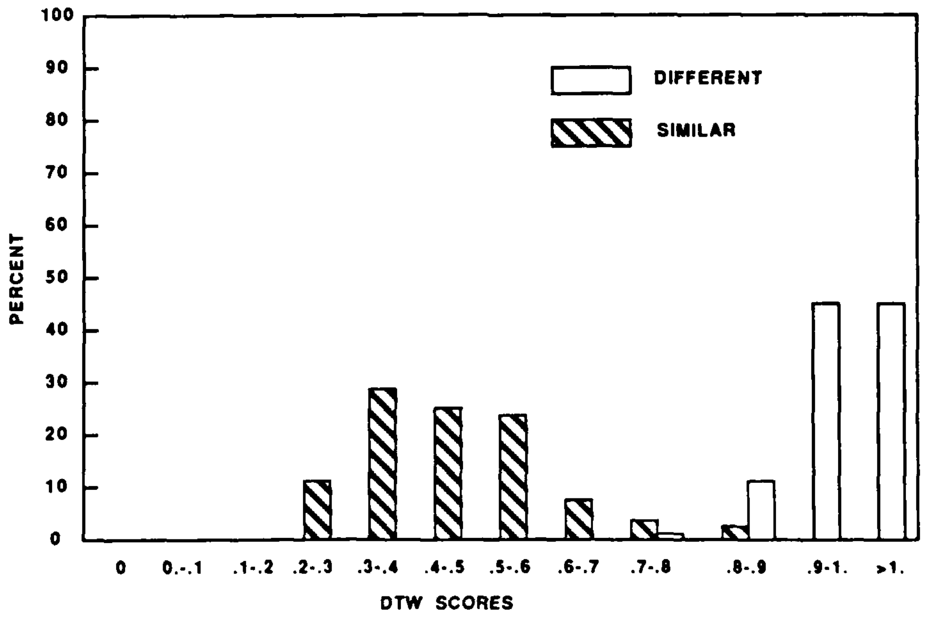
Figure 3.14. DTW Scores with Pre-Emphasis: μ = .95, 10 Bit Samples, PORDER = 16

MCIA-205



a. RNS {23,11,2}

MCIA-206



b. RNS {13,7,5}

Figure 3.15. DTW Scores with Pre-Emphasis: μ = .95, 10 Bit Samples, PORDER = 20

MCIA-265

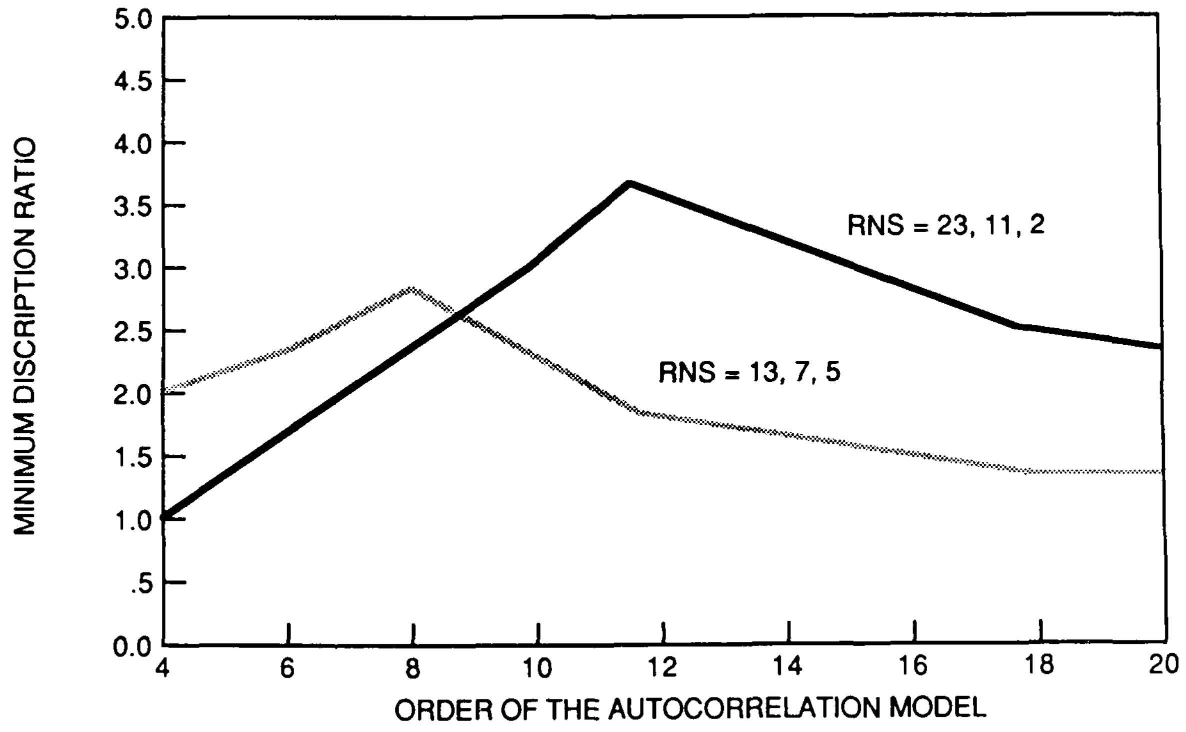


Figure 3.16. Minimum Discrimination Ratios vs. Autocorrelation Order

MCIA-266

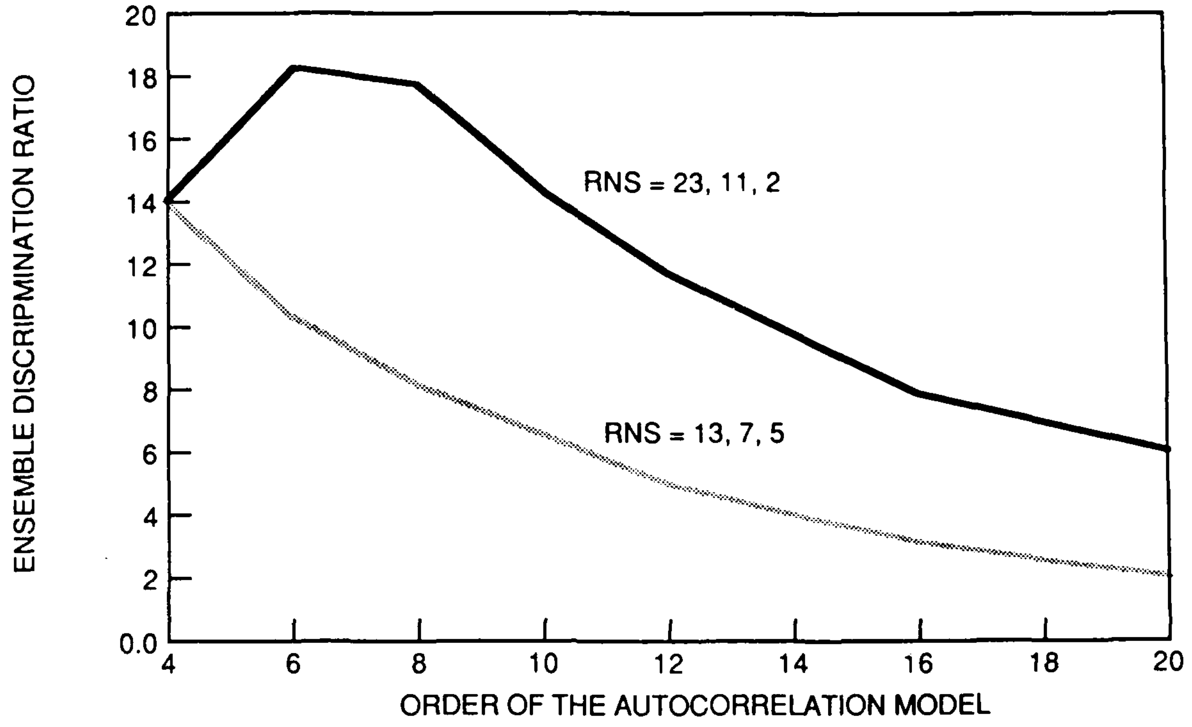


Figure 3.17. Ensemble Discrimination Ratios vs. Autocorrelation Order

To summarize these results, PORDER = 12 seems to give best recognition results among the values tested, but 10 and 8 seem to be adequate for good word recognizability. They also offer a savings in library space, processing time, and the range required for the autocorrelation analysis if done in RNS.

3.6 RECOGNITION PERFORMANCE IN NOISE

The RNS implementation of a DTW-based speech recognition system performed well even for quite coarse speech input quantization levels. This led us to consider whether this implementation might perform well in the presence of additive noise in the speech sample. To study the performance in a noisy environment, normally distributed (zero-mean) noise random variables were added to the test input speech samples following utterance detection.

3.6.1 Noise Model

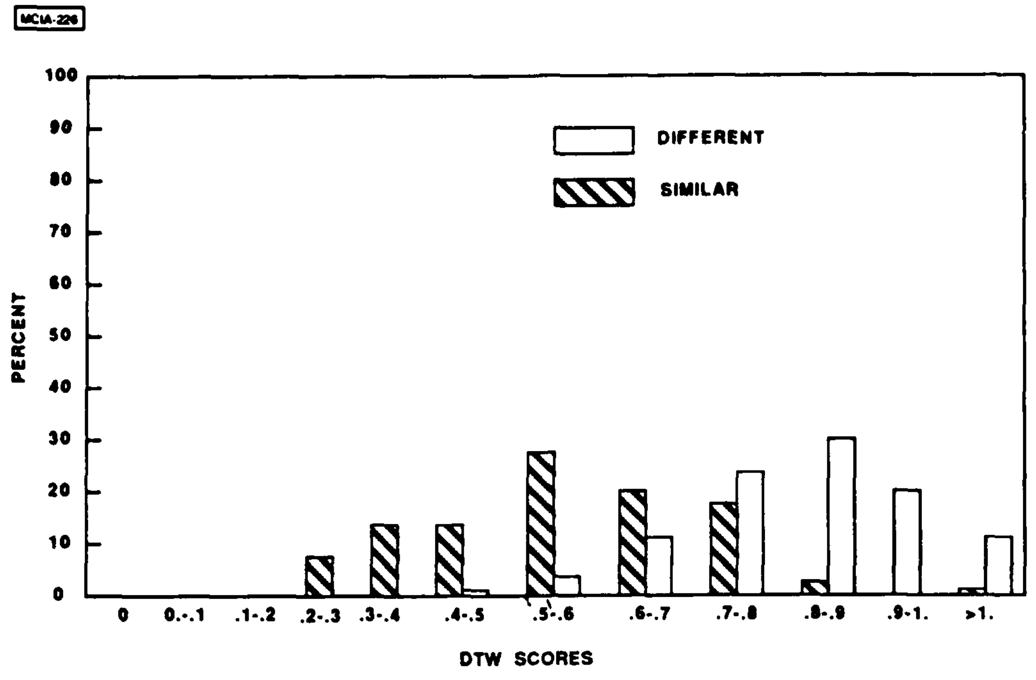
Following detection of an utterance in the test speech input data stream, the sample variance is calculated for the set of samples comprising the utterance. This determines the noise standard deviation, s_n, used to obtain the desired signal-to-noise ratio (SNR). Uniformly distributed integer variables are converted to floating point representation and used to construct normally distributed random variables, with zero mean and standard deviation one, by means of Knuth's algorithm P [8]. Premultiplication by s_n yields normally distributed random variables with zero mean and standard deviation, s_n. These are added to the speech input samples of the utterance, giving the desired SNR.

If pre-emphasis is selected (not advisable when there is much noise present), it is applied after the addition of the noise variables. Normalization of the test samples then proceeds, followed by the usual autocorrelation analysis to produce the test autocorrelation vectors used in the computation of local distortions.

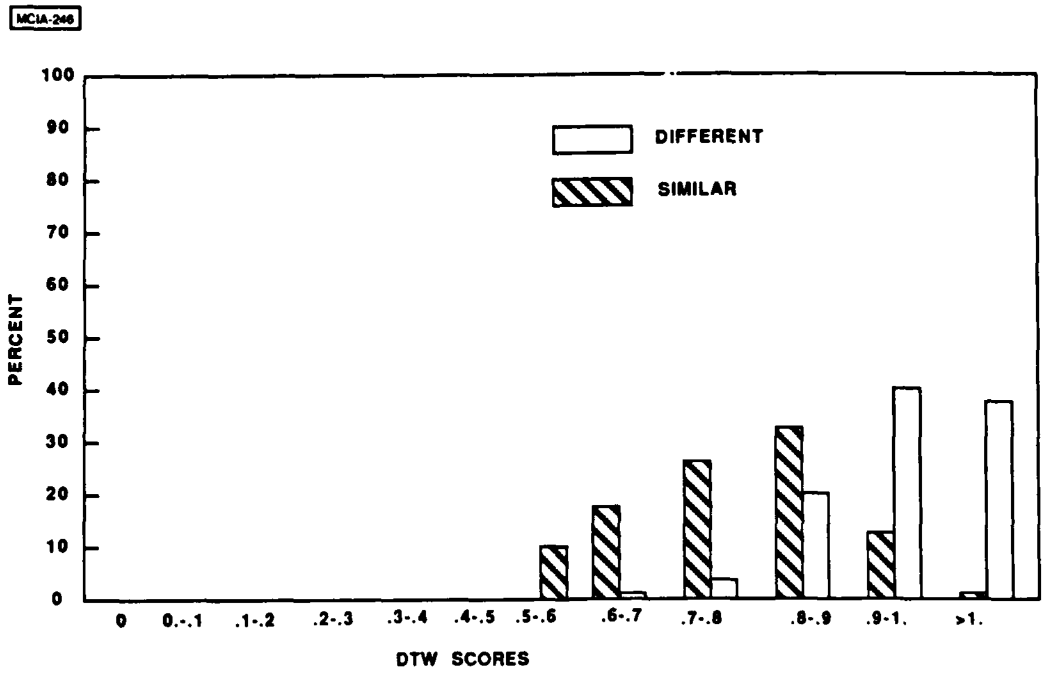
3.6.2 Simulation Results

In this section we present results from simulations run for several SNRs: 10, 15, 20, and 30 dB. Simulation runs have been made both with and without pre-emphasis of the noise-corrupted speech samples. Addition of noise shifts the DTW score histograms further to the right. If much noise is added, pre-emphasis becomes inappropriate because the noise dominates the residuals left from the first-order differencing. This is seen in figures 3.18 and 3.19 for simulations with and without pre-emphasis, respectively, with additive noise introduced at a 10 dB SNR. (Compare these to figures 3.2 and 3.3 for the noiseless case.) Pre-emphasis of the speech samples, formerly very helpful, is now destructive. In the absence of pre-emphasis, some separability still remains for the RNS {13,7,5} with lower threshold (figure 3.19b). (Only a single recognition error resulted for this case; it is the same single error, recognition of "three" as "two," that resulted in the noiseless case of figure 3.3b.)

With the higher SNRs 15 dB (figures 3.20 and 3.21) and 20 dB (figures 3.22 and 3.23), better separability is retained between histograms of scores for matches of similar words and those for matches of different words. At 15 dB SNR, recognition is still better if pre-emphasis is not employed, but pre-emphasis of the speech samples may improve the separability between histograms at 20 dB SNR.

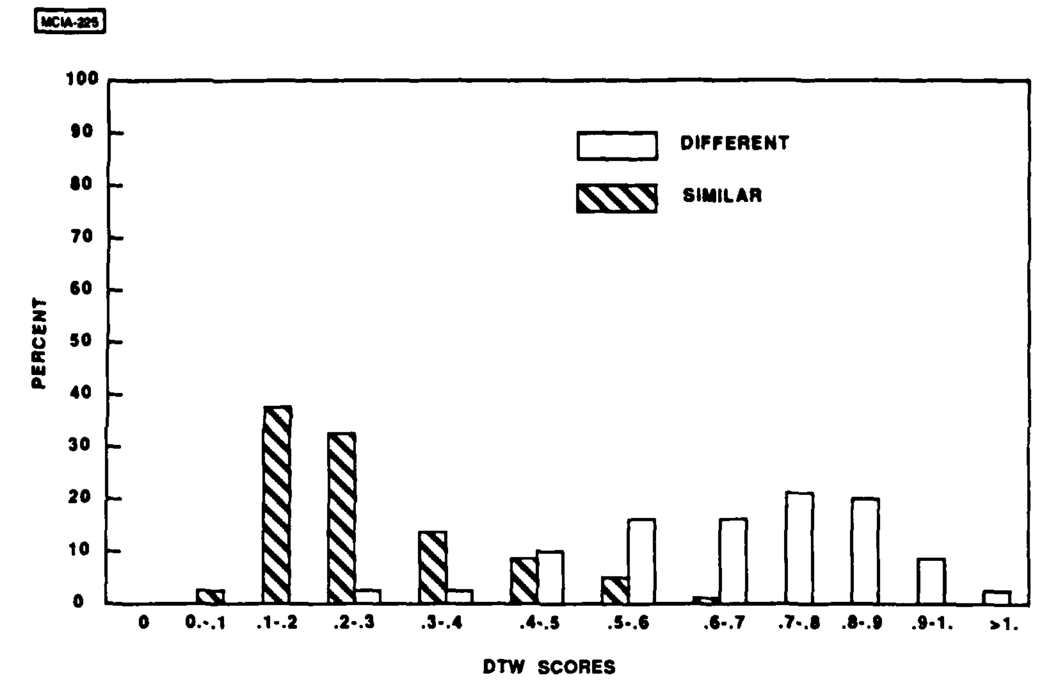


a. RNS {23,11,2}

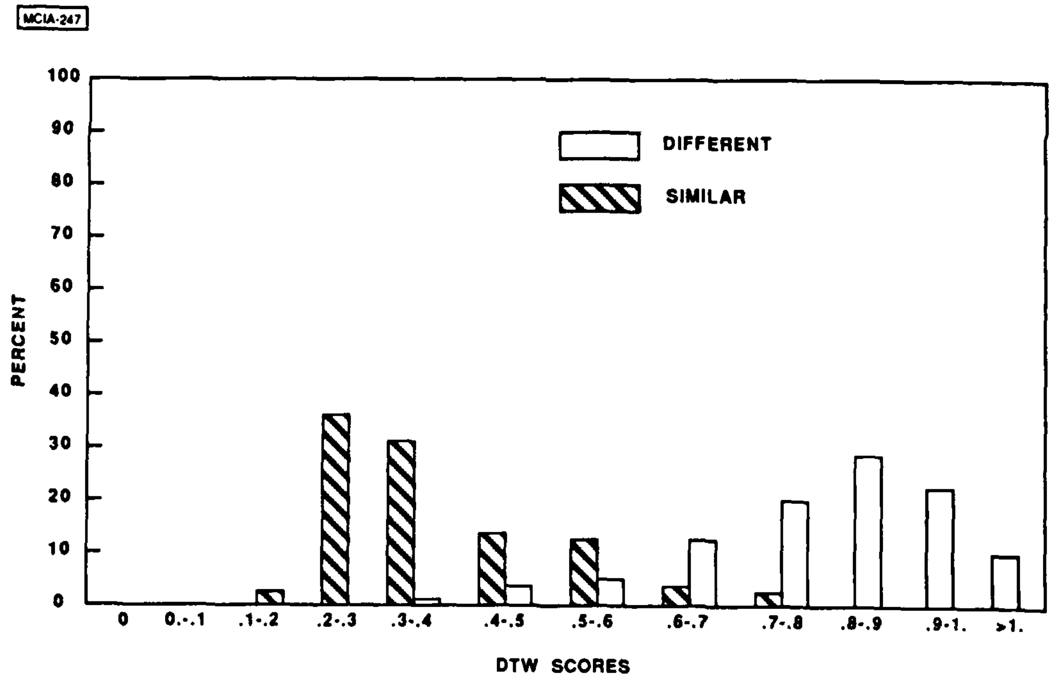


b. RNS {13,7,5}

Figure 3.18. DTW Scores with Pre-Emphasis: μ = .95, 10 Bit Samples, Normally Distributed Noise SNR = 10 dB

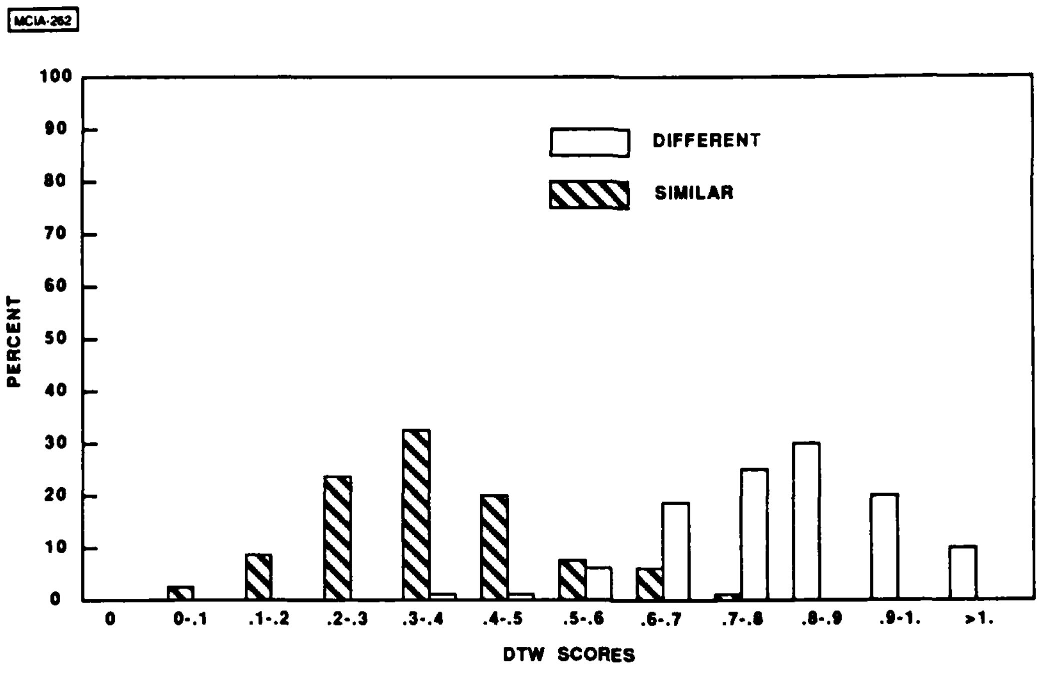


a. RNS {23,11,2}

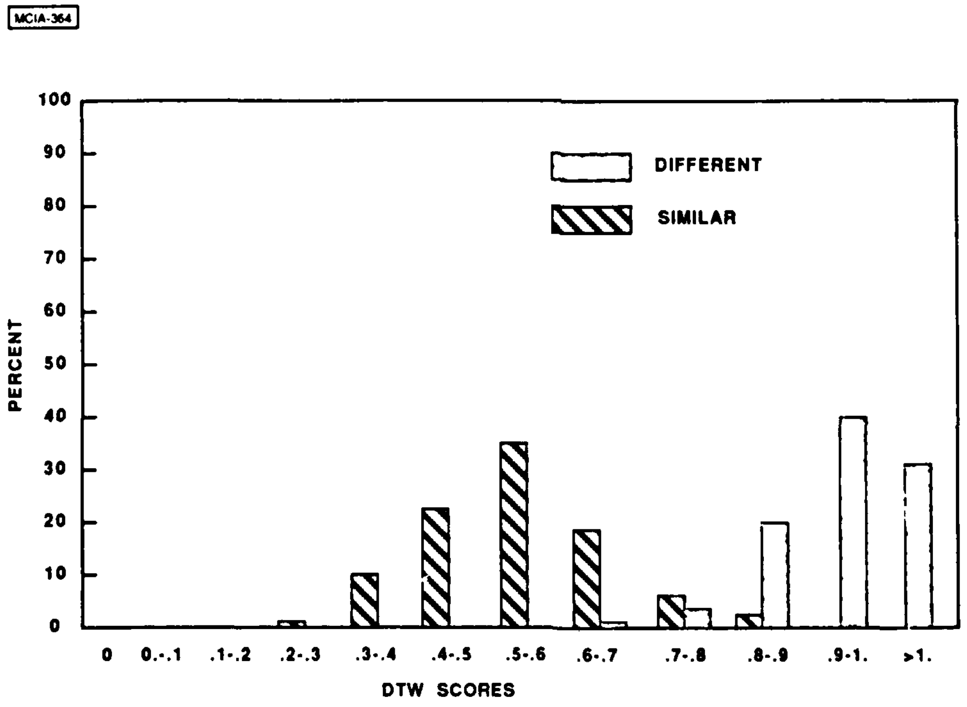


b. RNS {13,7,5}

Figure 3.19. DTW Scores without Pre-Emphasis: $\mu = .95$, 10 Bit Samples, Normally Distributed Noise SNR = 10 dB

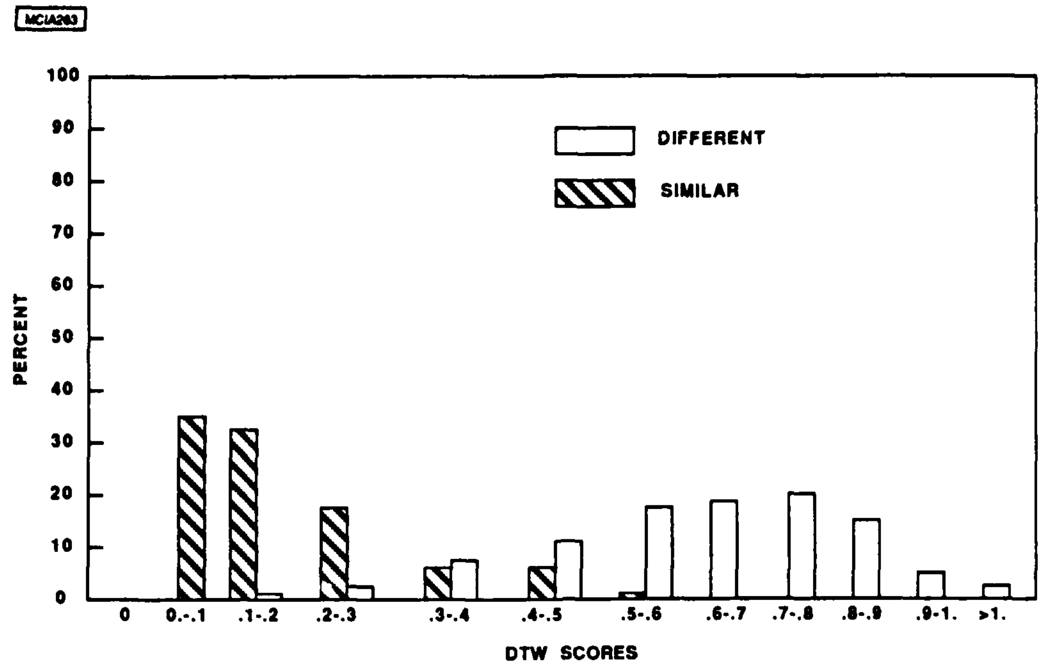


a. RNS {23,11,2}

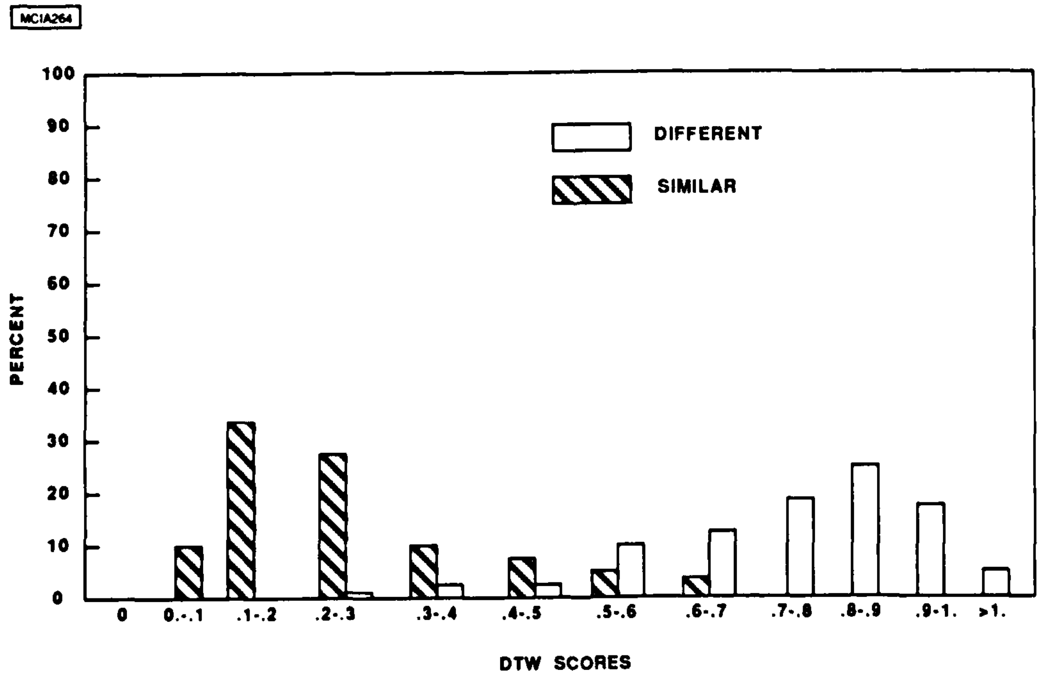


b. RNS {13,7,5}

Figure 3.20. DTW Scores with Pre-Emphasis: μ = .95, 10 Bit Samples, Normally Distributed Noise SNR = 15 dB

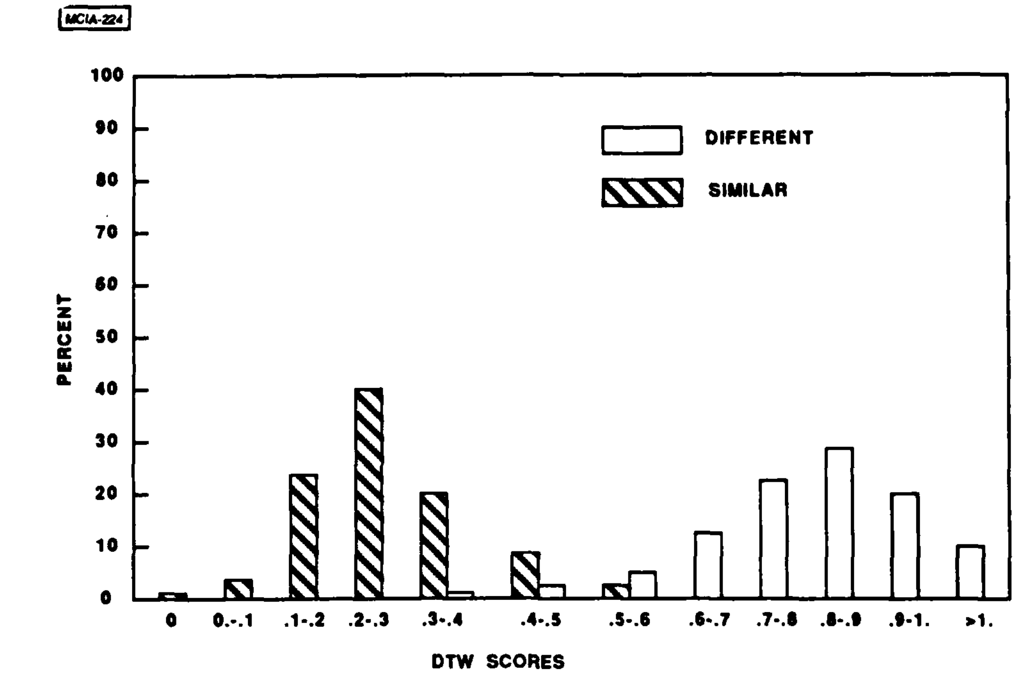


a. RNS {23,11,2}

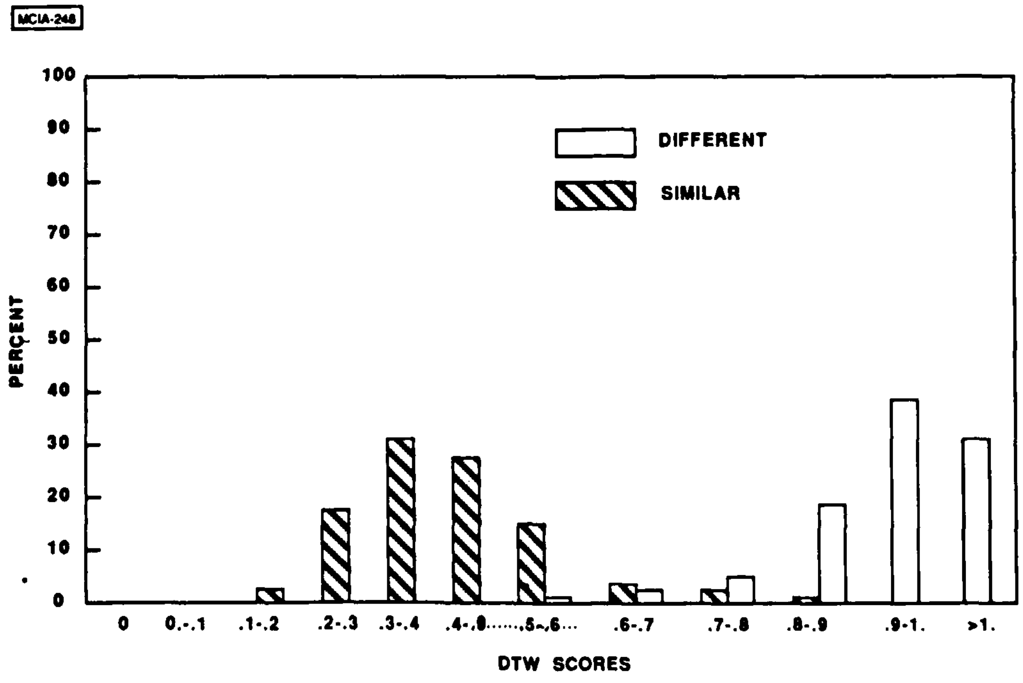


b. RNS {13,7,5}

Figure 3.21. DTW Scores without Pre-Emphasis: μ = .95, 10 Bit Samples, Normally Distributed Noise SNR = 15 dB

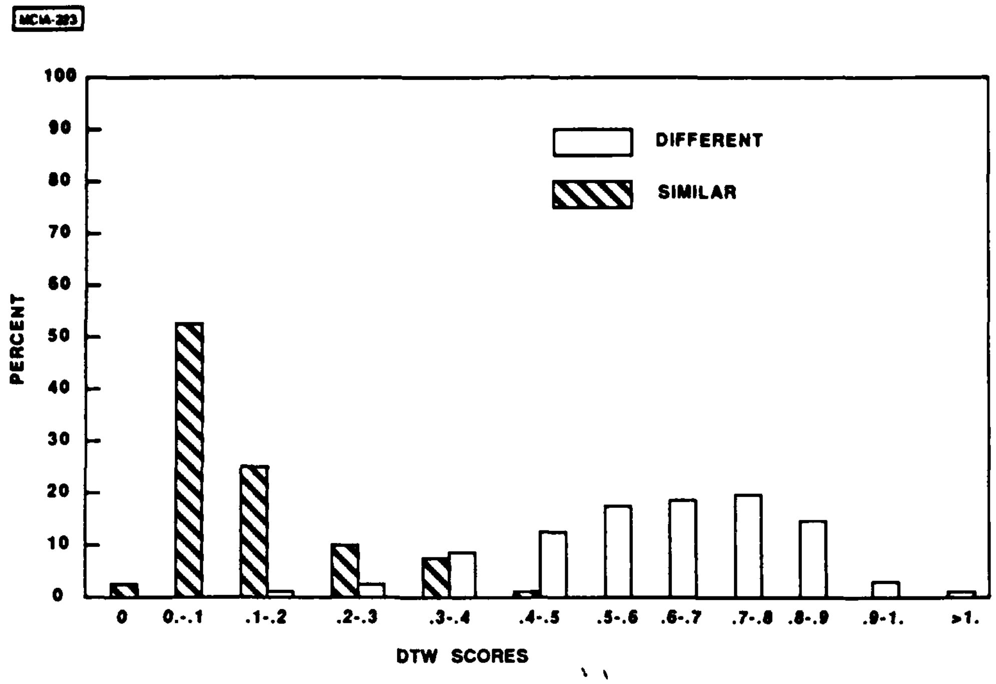


a. RNS {23,11,2}

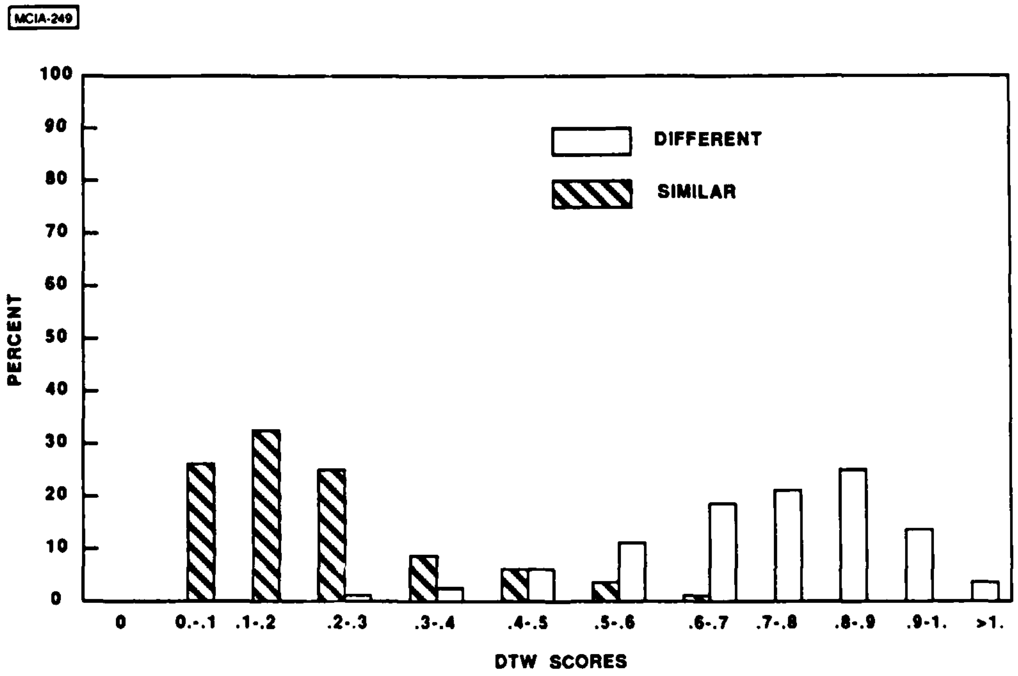


b. RNS {13,7,5}

Figure 3.22. DTW Scores with Pre-Emphasis: μ = .95, 10 Bit Samples, Normally Distributed Noise SNR = 20 dB



a. RNS {23,11,2}



b. RNS {13,7,5}

Figure 3.23. DTW Scores without Pre-Emphasis: μ = .95, 10 Bit Samples, Normally Distributed Noise SNR = 20 dB

When the SNR is 30 dB, the separations between score histograms (figures 3.24 and 3.25) are comparable to those obtained when no noise is present (figures 3.2 and 3.3). Pre-emphasis again is definitely helpful; no recognition errors resulted when pre-emphasis was used.

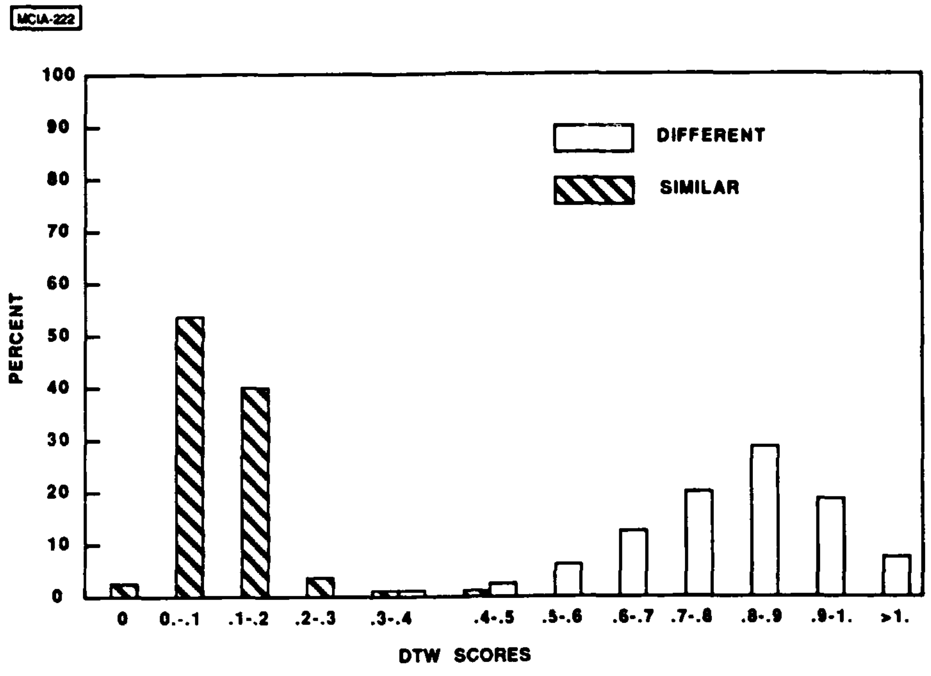
Table 3.2 lists the number of recognition errors counted in the simulations versus the signal-to-noise ratio applied. Four columns, corresponding to the four cases (with or without pre-emphasis, RNS = {23,11,2} or {13,7,5}) are shown.

Table 3.2

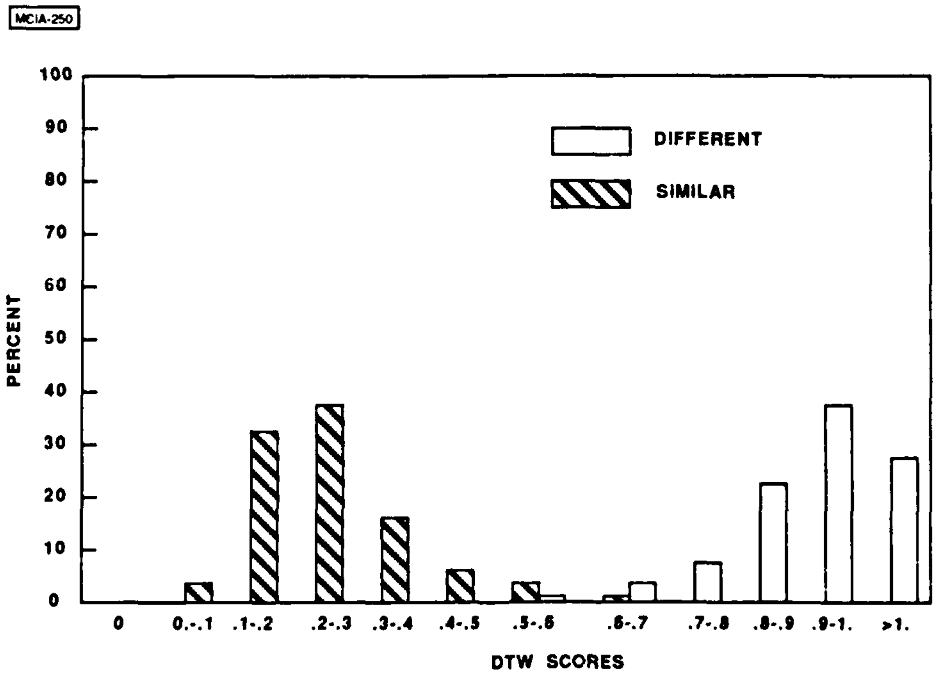
Error Analysis for Performance in Noise
Number of Recognition Errors

	with pre-emphasis		without pre-emphasis	
SNR/ RNS	{23,11,2}	{13,7,5}	{23,11,2}	{13,7,5}
10 dB	16	18	3	1
15 dB	9	3	1	1
20 dB	3	0	1	1
30 dB	0	0	0	1
∞	0	0	0	1

To summarize these results, the introduction of additive noise produces overlaps in the score histograms for matches of similar words and matches of different words. For the RNS {13,7,5} without pre-emphasis, some reasonable separation is retained at an SNR of

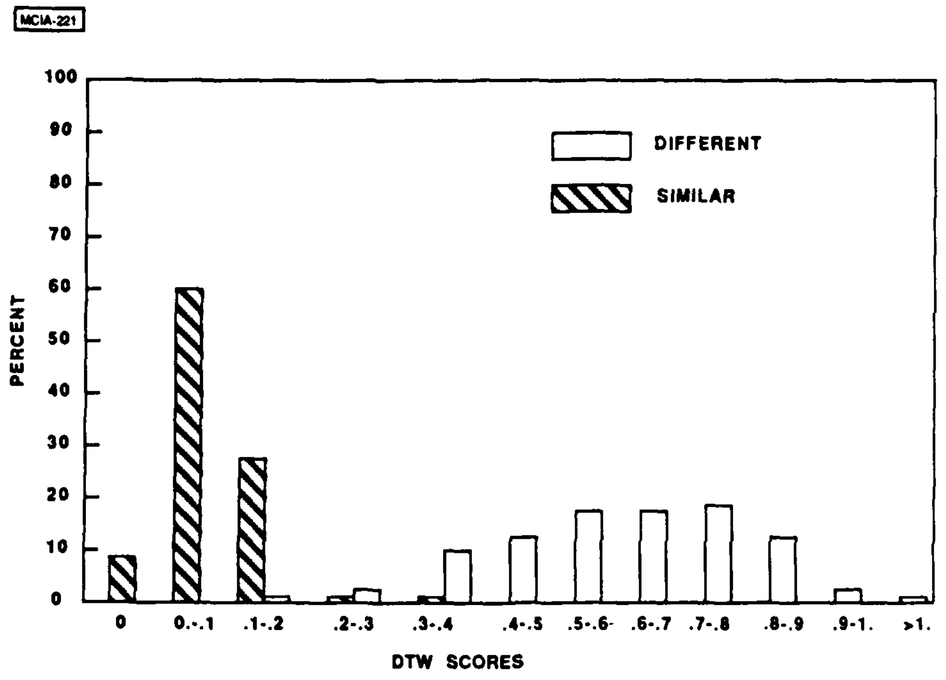


a. RNS {23,11,2}

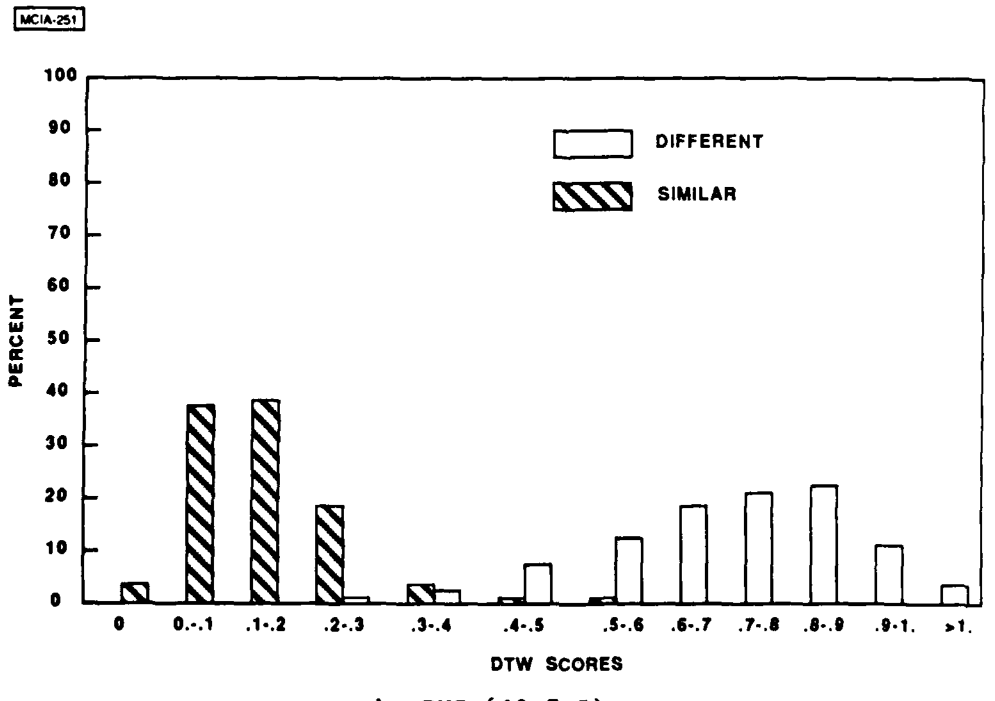


b. RNS {13,7,5}

Figure 3.24. DTW Scores with Pre-Emphasis: μ = .95, 10 Bit Samples, Normally Distributed Noise SNR = 30 dB



a. RNS {23,11,2}



b. RNS {13,7,5}

Figure 3.25. DTW Scores without Pre-Emphasis: μ = .95, 10 Bit Samples, Normally Distributed Noise SNR = 30 dB

10 dB or better, and recognition performance, while deteriorating, is better than might be expected, especially in light of the marginal performance of LPC models in noise.

3.7 COMPARISON OF SINGLE-SPEAKER AND MULTISPEAKER PERFORMANCE

Multispeaker recognition is difficult with the present configuration of the speech recognition system. This is, in part, because utterance detection has been treated strictly as a preprocessing function of the system. All normalization of sample values to compensate for variations in speaker dynamics, microphone placement, system gain differences, etc., takes place after an utterance has been detected and its beginning and endpoints have been defined. In actual practice, however, some normalization may be required before the utterance endpoints are fixed. Otherwise, there can be considerable variation in the length of an utterance of the same word from one speaker to another or from one recording session to the next. Normalization that takes place after utterance detection does not correct these discrepancies in utterance length.

Table 3.3 gives the utterance lengths in segments or frames for two testsets produced by speakers different from speaker A, who produced the database (table 3.1). Comparison of the lengths shows some striking discrepancies between productions of table 3.3 and those of table 3.1. For some testset words, e.g., "four," "five," and "foxtrot," there are few or no counterparts in the library produced by speaker A comparable in length to the testset utterances of speakers B and C. For this reason we have loosened the global constraint, g, set at 9 in all previous simulations to larger values for simulations with different speakers.

Table 3.3

Testset Utterance Lengths (in Frames) for Speakers B and C

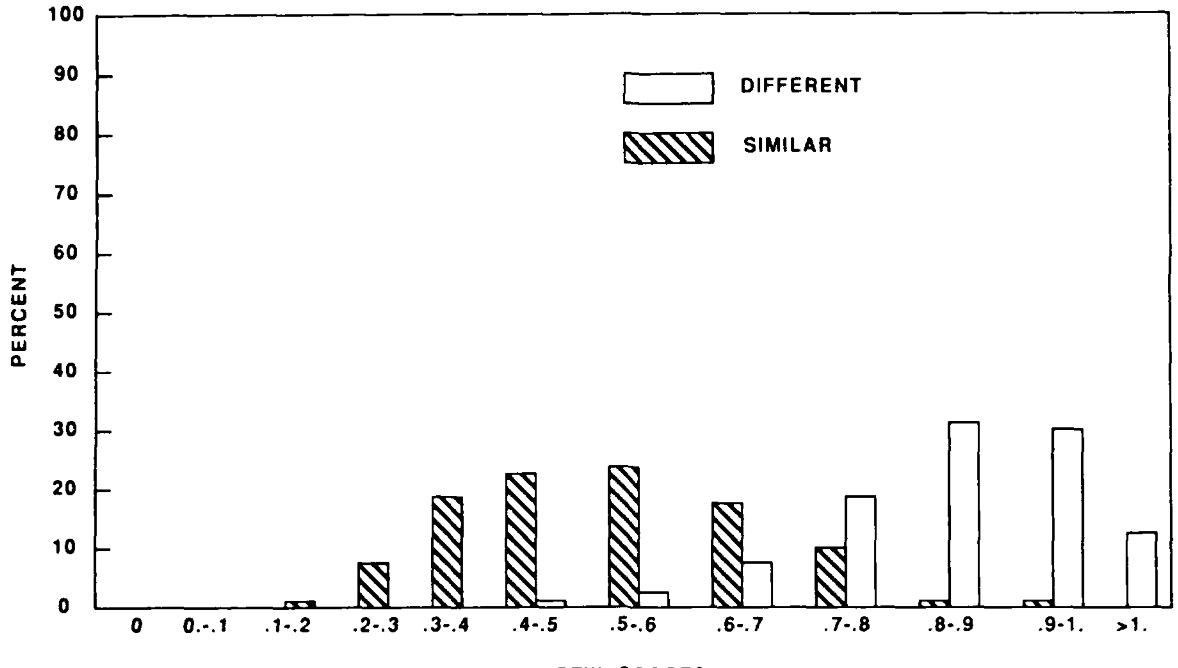
Word	Testset B	Testset C
alpha	50	51
bravo	62	61
charlie	58	57
delta	53	49
echo	46	47
eight	44	47
five	64	50
four	43	39
foxtrot	62	75
golf	36	57
hotel	61	62
india	62	55
juliet	69	81
kilo	52	62
lima	54	55
mike	49	53
nine	65	64
november	77	79
oh	58	44
one	47	51
oscar	58	65
papa	61	56
quebec	57	60
romeo	65	76
seven	46	58
sierra	53	74
six	60	81
tango	54	67
three	48	49
two	46	44
uniform	75	90
victor	52	69
whiskey	55	59
xray	63	72
yankee	61	63
zero	69	58
zulu	73	61

Simulation runs were performed using the RNS {23,11,2} with input quantization at 10 bits for each of the testsets for speakers B and C with the global constraint set at the values 9, 12, 15, and 18. The histograms of DTW scores corresponding to matches of similar words and matches of different words are shown in figures 3.26 - 3.29 for the respective settings of the global constraint. In each case, the a plot shows the results for speaker B, and the b plot shows scores for speaker C.

The DTW scores are relatively high for all cases, and there is considerable overlap between histograms of scores for matches of similar words and those for matches of different words, making correct identification difficult. This probably reflects, in part, the decision to treat utterance detection as a preprocessing function. Better separation would be expected if normalization were applied prior to the utterance detection. Other measures that might improve separation include the use of more than one speaker in the database used for constructing libraries, and relaxation of the endpoint constraints used in finding the DTW path. (The endpoint constraints applied in all simulations require the path to pass through the points (1,1) and (m,n), thus insisting that the first test and reference frames be matched and the last test and reference frames be matched. This constraint may be inappropriate in situations where the utterance endpoints vary considerably from speaker to speaker.)

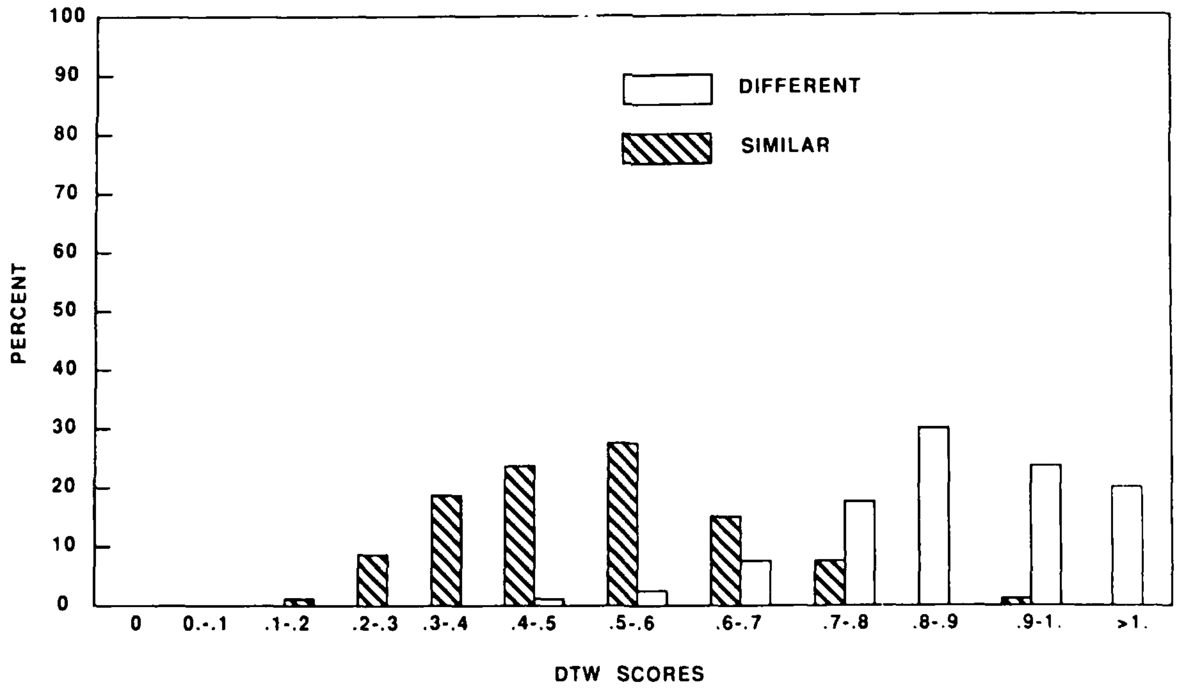
Table 3.4 contains a summary of the error analysis for these simulation runs. The column for testset B (or C) shows the number of recognition errors resulting when the testset of speaker B (or C) is used with the library constructed from utterances spoken by speaker A.

MCIA-327



a. RNS {23,11,2}
Speaker B

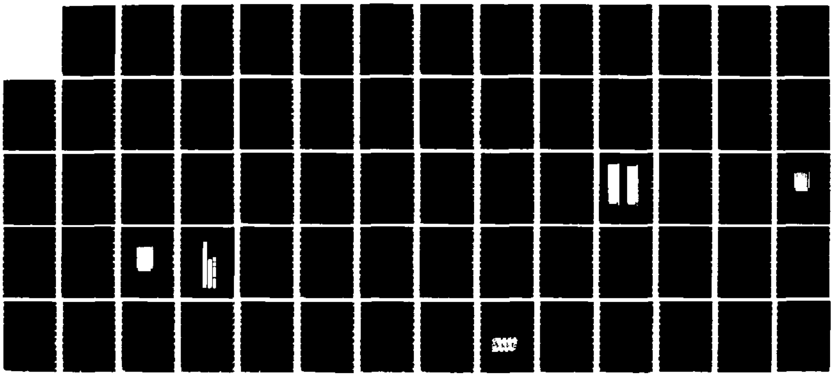
MCIA-328

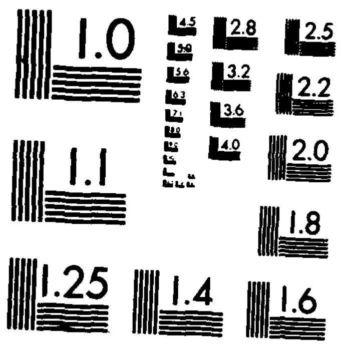


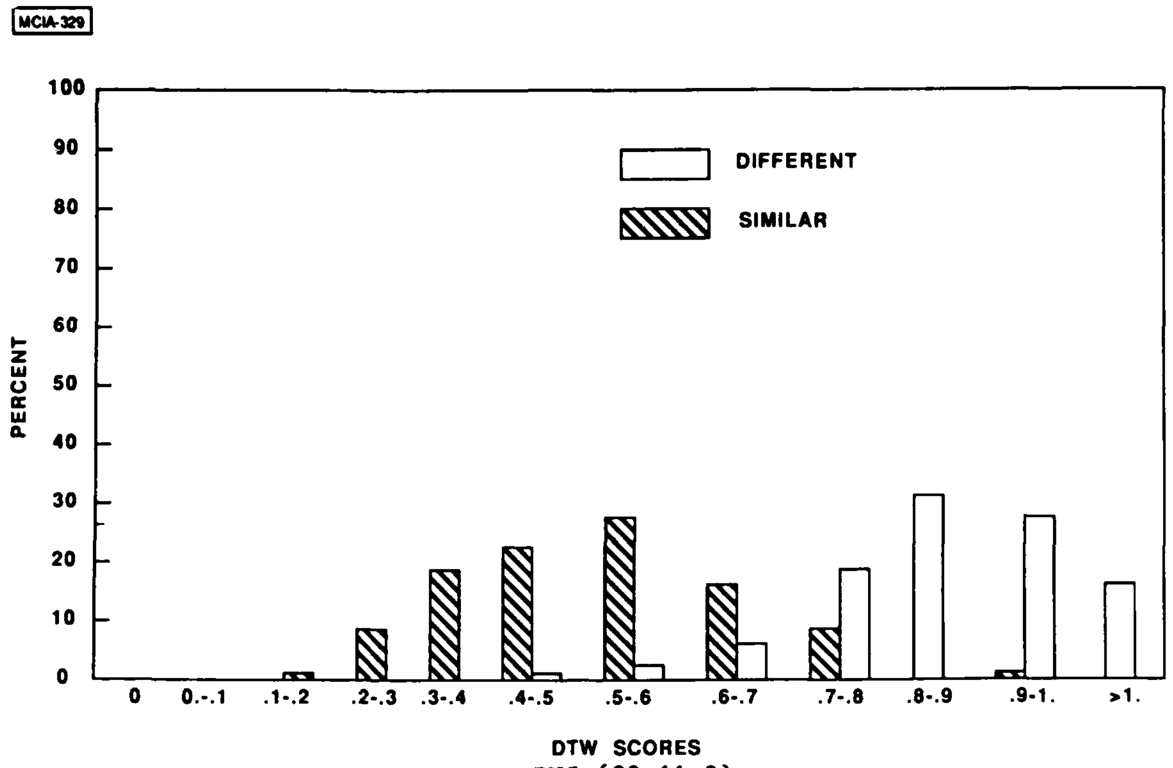
b. RNS {13,7,5}
Speaker C

Figure 3.26. DTW Scores with Pre-Emphasis: μ = .95,
Global = 9

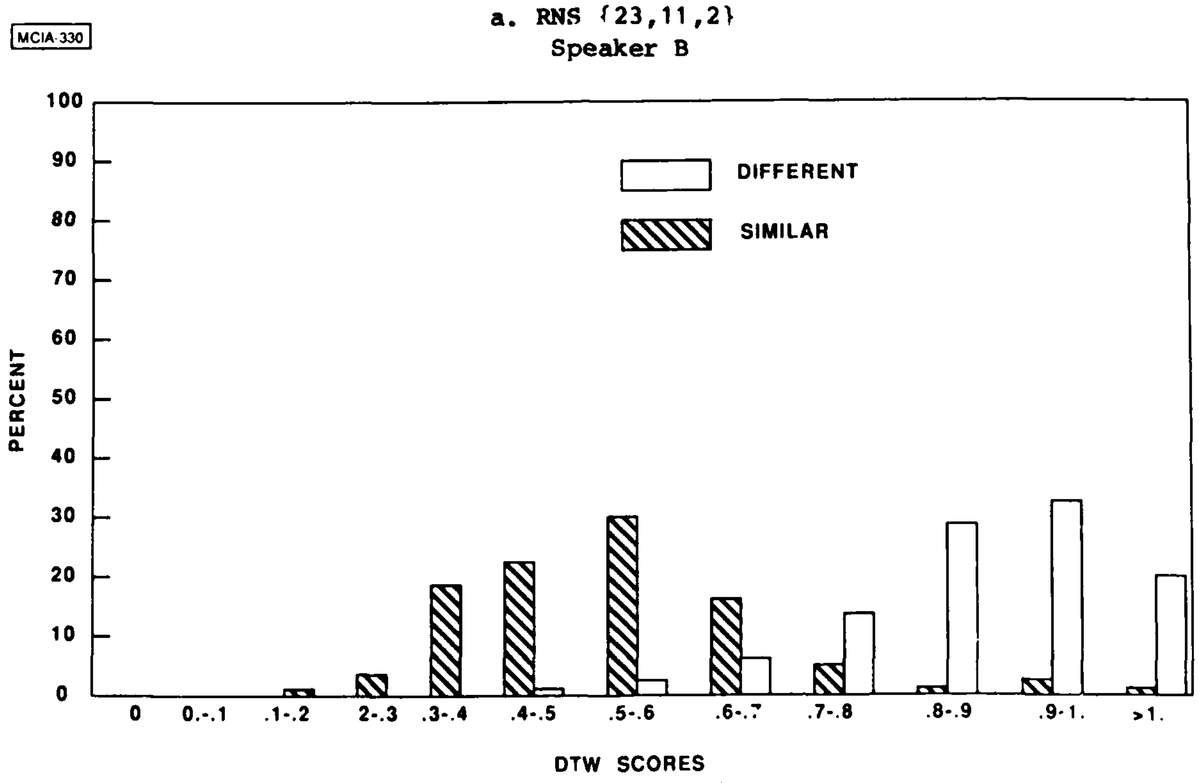
AD-A173 600 ADVANCED ARCHITECTURES FOR SIGNAL PROCESSORS (A SIMPLE 2/2
RNS (RESIDUE NUMB (U) MITRE CORP BEDFORD MA
D O CARHOUN ET AL JUL 86 MTR-9727 RADC-TR-86-59
UNCLASSIFIED F19628-84-C-0001 F/G 9/2 NL




1.0
4.5
5.0
5.6
6.3
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6



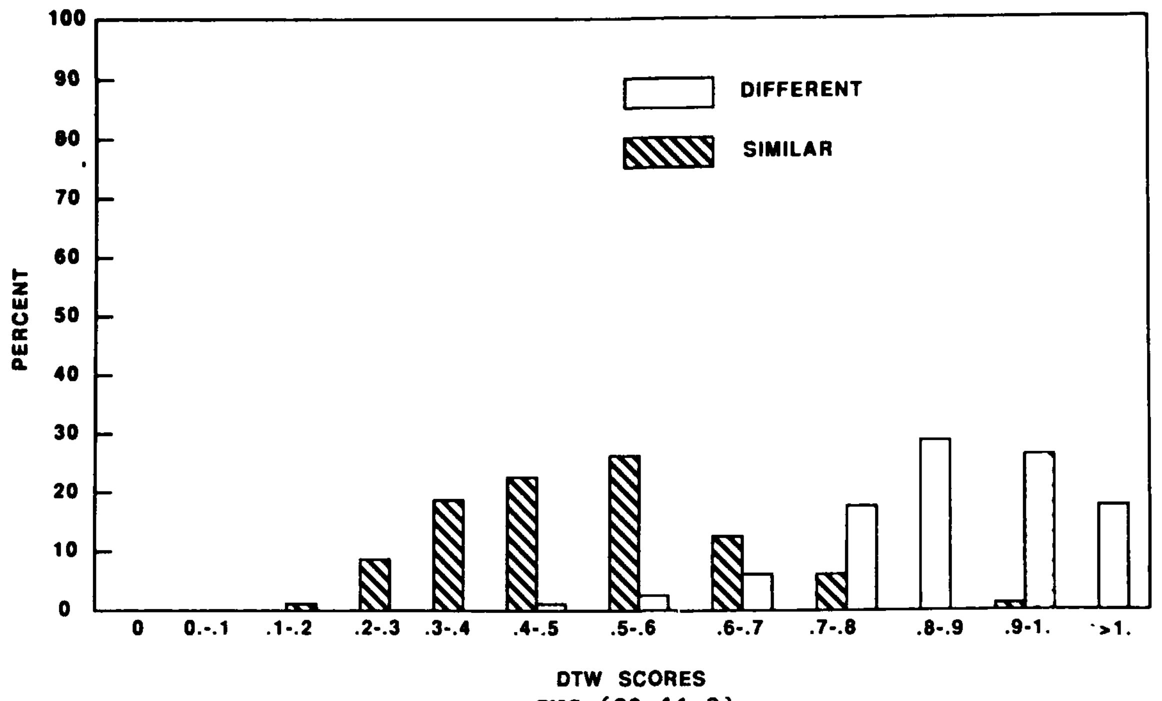
a. RNS {23,11,2}
Speaker B



b. RNS {13,7,5}
Speaker C

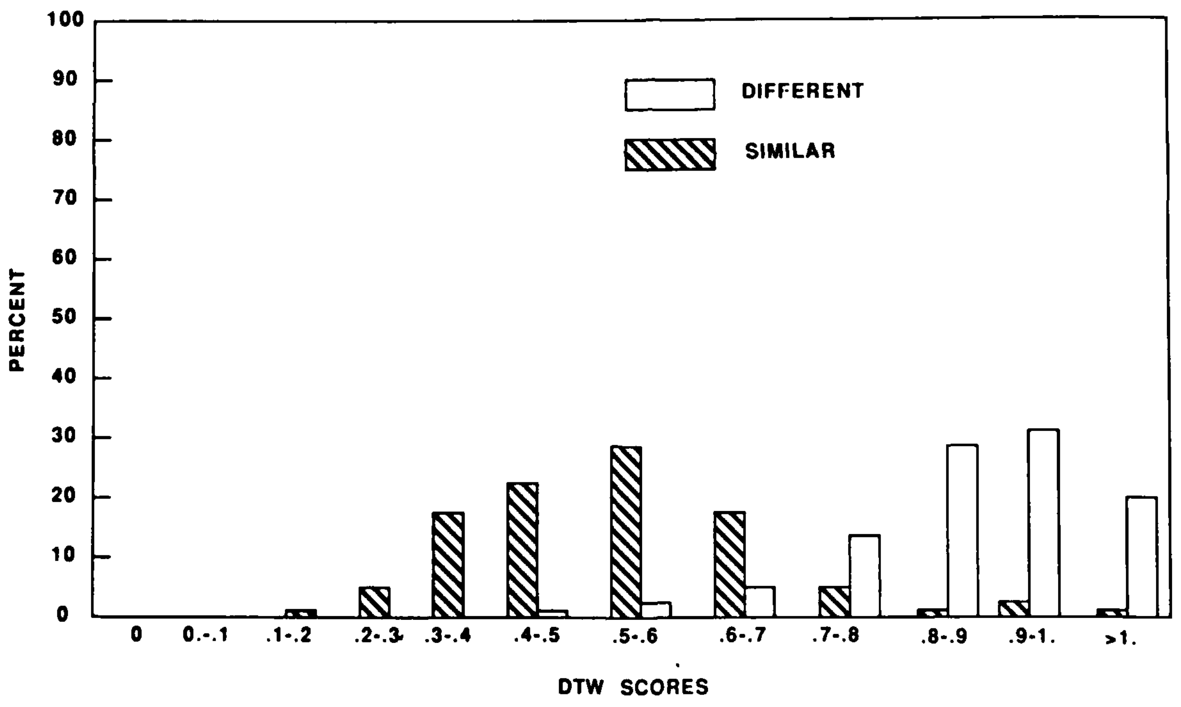
Figure 3.27. DTW Scores with Pre-Emphasis: μ = .95, Global = 12

MCIA-331



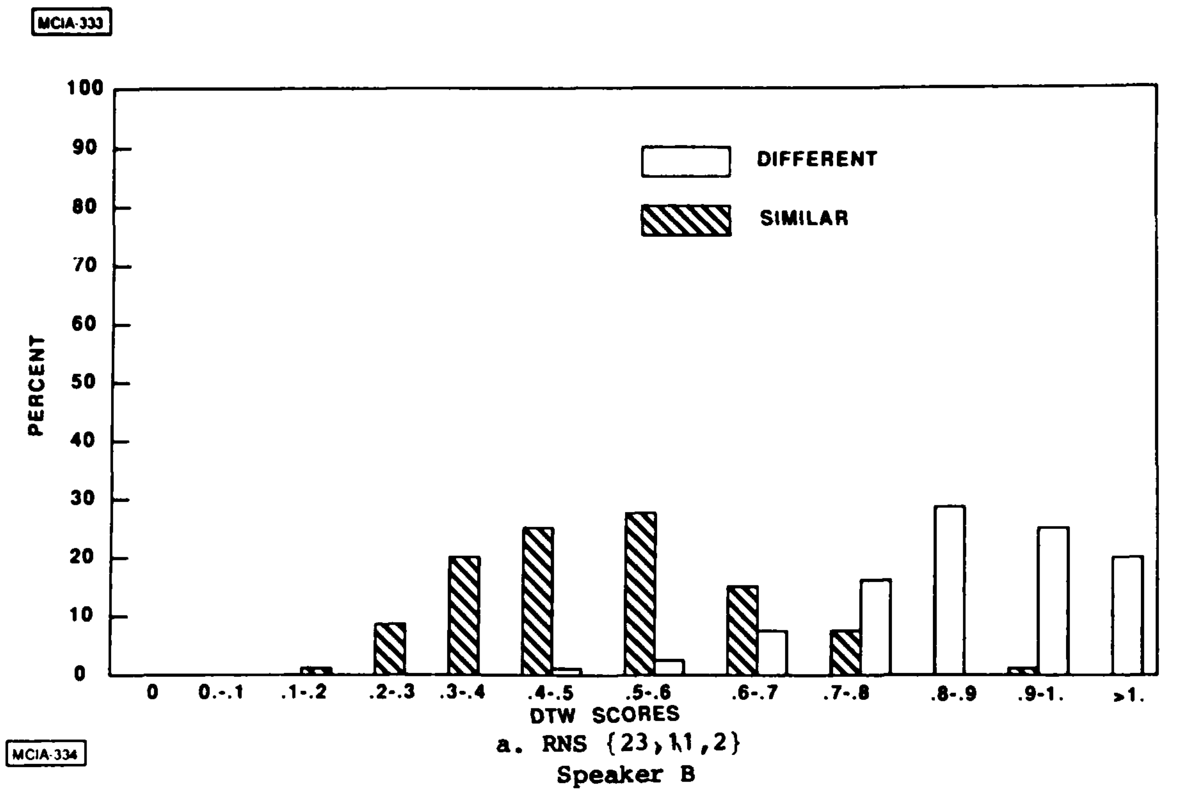
a. RNS {23,11,2}
Speaker B

MCIA-332

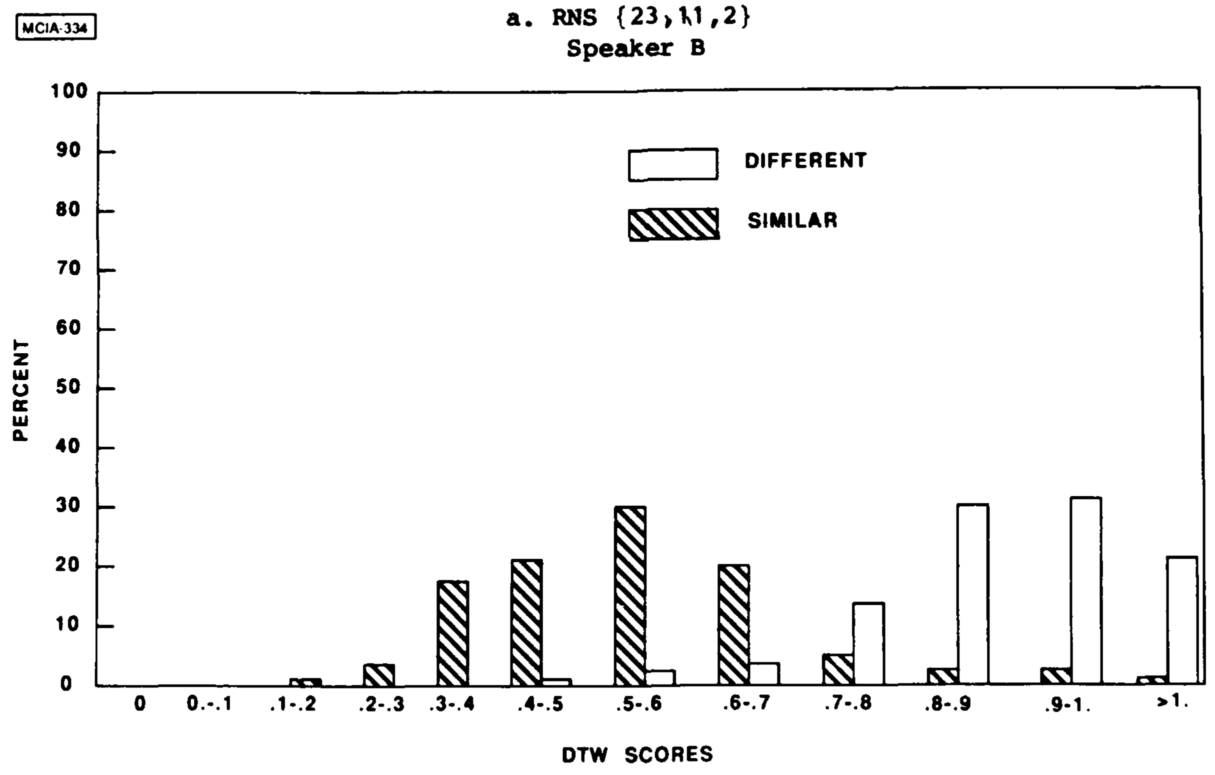


b. RNS {13,7,5}
Speaker C

Figure 3.28. DTW Scores with Pre-Emphasis: μ = .95, Global = 15



a. RNS {23,11,2}
Speaker B



b. RNS {13,7,5}
Speaker C

Figure 3.29. DTW Scores with Pre-Emphasis: μ = .95,
Global = 18

Table 3.4

Number of Recognition Errors for Multispeaker Recognition

Global Constraint	Testset B	Testset C
9	7	13
12	8	10
15	7	9
18	6	7

3.8 DTW PATH COMPUTATIONS AND THE EFFECTS OF OVERFLOWS

As discussed in section 2.8.3, the DTW path computations can be successfully performed in RNS if the local distortion values are first quantized to two or a few levels. The revised shortest-path computation was given in section 2.8.3. Two types of potential overflow must be considered. First, the magnitude comparisons of steps 3 and 4 are to be performed in the largest residue channel. An overflow results if the difference, in absolute value, between the cumulative path distances under comparison exceeds half the largest modulus employed. Second, the cumulative distances themselves are represented by residues in all channels used. An overflow results if the cumulative distance for the presumed best path exceeds the range of the RNS. This is a serious error, generally leading to a recognition error, for the resulting path score is much lower than its true value.

Let us consider the second and more serious overflow possibility. If we use the weight function of equation (8), the cumulative cost for any path cannot exceed the number of reference frames. Since the longest utterance in our test library consists of 104 frames, an RNS of range 104 or greater suffices for the shortest-path computation (under one-bit quantization of distortion values). Hence, for an RNS composed of the two prime moduli {13,11} no overflows of this type can occur, but for an RNS composed of the two moduli {11,7}, overflow may result.

The first and less serious type of overflow is much more likely to occur, and care should be taken to ensure that the first residue channel is of sufficient size. Table 3.5 shows the number of errors of the first type resulting for various choices of two-modulus RNS for the DTW path computations. In all cases shown, the distortion-function computations were performed using an RNS composed of the three prime moduli {23,11,2}. Quantization of the distortion values was performed as in figure 2.6 to a single bit (match or no-match) using a threshold value $T = p_1 = 23$. The last column of the table shows the number of recognition errors for simulations performed using the 37-word testset.

For the last two RNS, {7,5} and {5,3}, the RNS range is exceeded much of the time by the cumulative distances. The results displayed in table 3.5 support the hypothesis that little harm results from occasionally overflowing the largest modulus in the path comparisons, provided that the number of overflows is not excessive. No degradation in recognition performance was observed until the larger modulus was reduced to 11, when the number of overflows exceeded 9000. No overflows were observed when the larger modulus was 19 or greater.

Table 3.5

Number of Overflows of First Modulus and Recognition Errors for Various RNS Choices for Path Computations with Two-Level Quantization of Distortion Values

Moduli	Number of Overflows	Recognition Errors
23,19	0	0
19,17	0	0
17,13	233	0
13,11	2,587	0
11, 7	9,113	1
7, 5	108,394	34
5, 3	265,530	36

For a hardware implementation the two largest moduli of the three employed for the distortion function calculation can be used to form an RNS for the DTW path calculations. The choice of RNS {23,11,2} for the distortion calculations is safe from this point of view; the range of the RNS composed from the moduli {23,11} is sufficient to avoid all serious overflows and to avoid less serious overflows most of the time. The choice of RNS {13,7,5}, however, entails more risk, as the range of the subset {13,7} may be insufficient. Overflow errors of the first type are likely to occur too frequently.

Table 3.6 presents similar results for three-level quantization of the distortion values (using the quantization shown in figure 2.6, with thresholds set at $p_1 = 23$ and $p_2p_1 = 253$). Again the RNS {23,11,2} was used for the distortion computations.

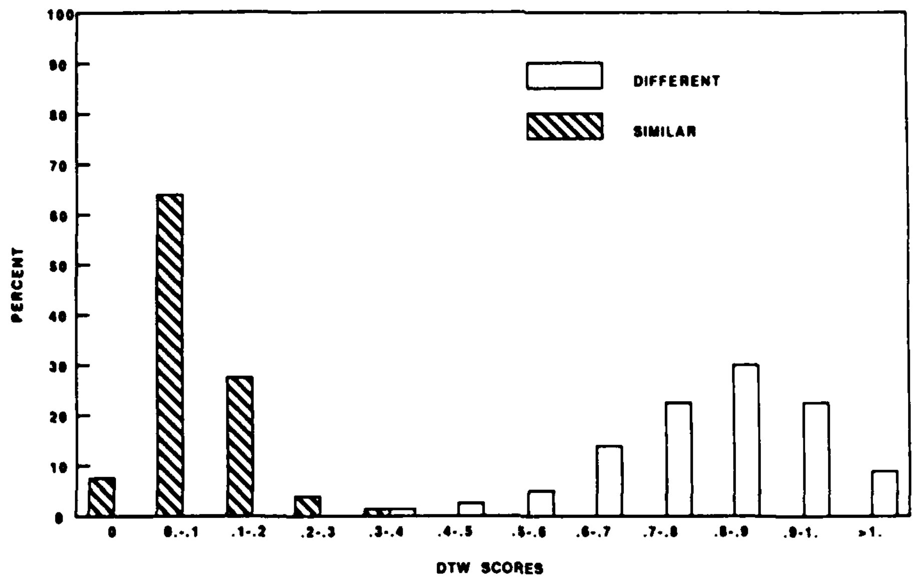
Table 3.6

Number of Overflows of First Modulus and Recognition Errors for Various RNS Choices for Path Computations with Three-Level Quantization of Distortion Values

Moduli	Number of Overflows	Recognition Errors
23,19	0	0
19,17	0	0
17,13	360	0
13,11	4,341	0
11, 7	15,188	1
7, 5	178,504	36
5, 3	344,788	36

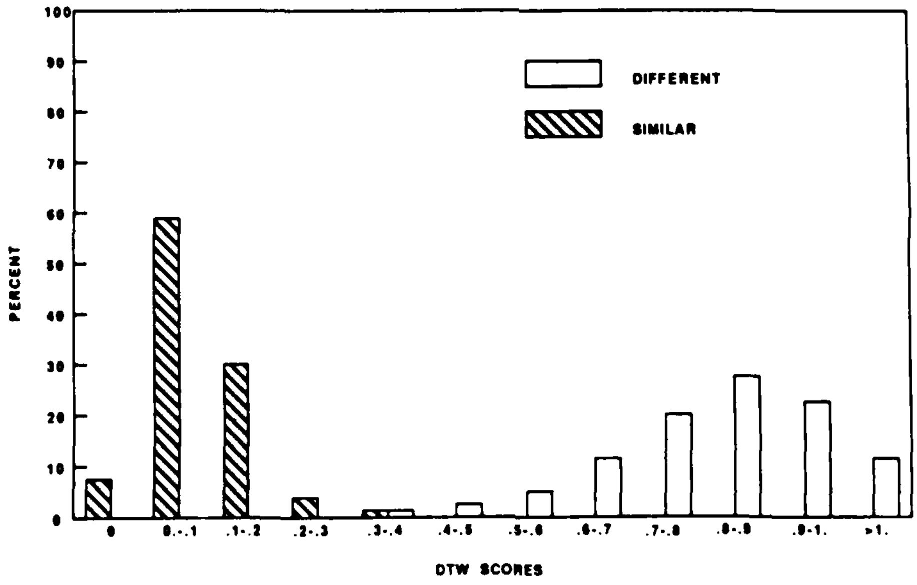
Histograms of DTW scores are shown for some of the simulations in figures 3.30 to 3.35. The a plots illustrate two-level quantization of the distortion values, and the b plots show scores obtained for three-level quantization. Figure 3.30a is identical to figure 3.2a. Figure 3.31a is similar: there is a slight shift of the histogram to the right for similar words. This shift becomes more pronounced as the second RNS is made smaller (figures 3.32a and 3.33a), but the histograms for matches of different words are largely unaffected. However, when the RNS is reduced to {7,5}, the effect of overflows in the cumulative distance calculation is felt, shifting the histogram of scores for matches of different words down to the left (figure 3.34a). When the RNS is reduced to the pair of moduli 5 and 3, both histograms are shifted far to the left, as almost all path scores have overflowed the small range (15) and been mapped into low values (figure 3.35a). Correct recognition is now out of the question.

MCIA-340



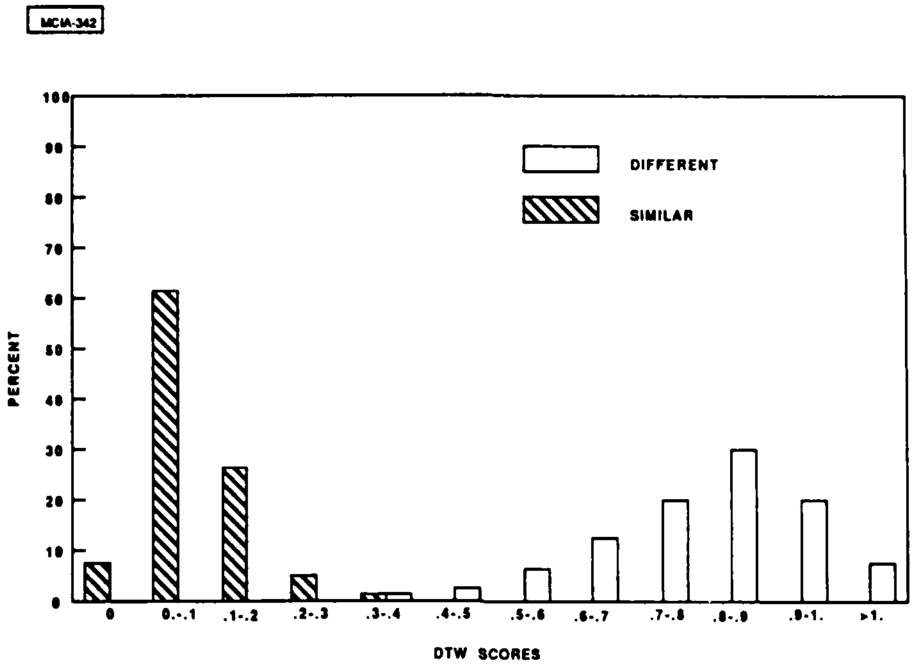
a. 2-Level Quantization

MCIA-341

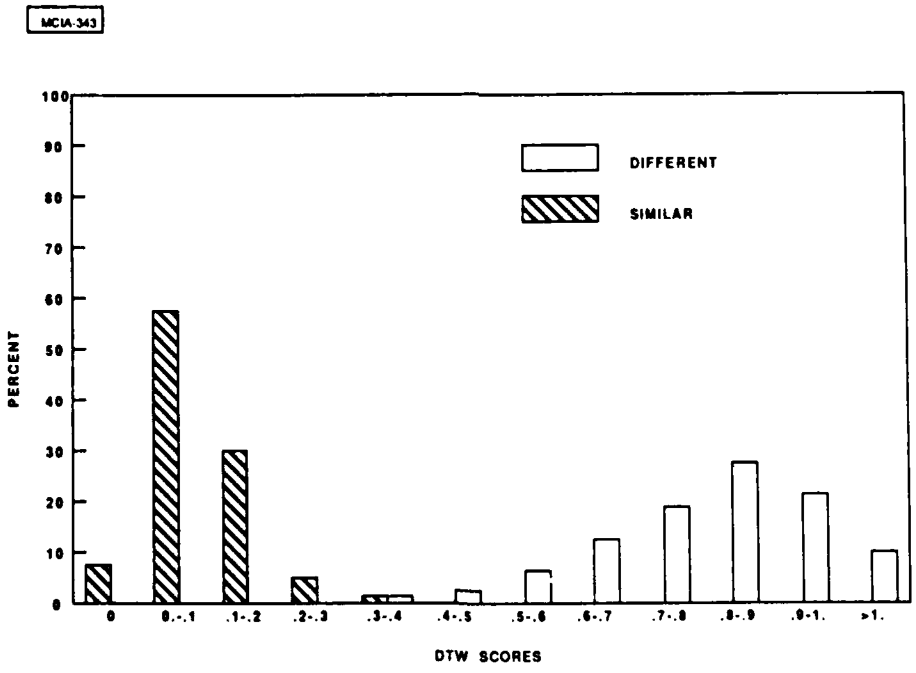


b. 3-Level Quantization

Figure 3.30. DTW Scores with Pre-Emphasis: μ = .95, RNS1: {23,11,2}, RNS2: {23,19}

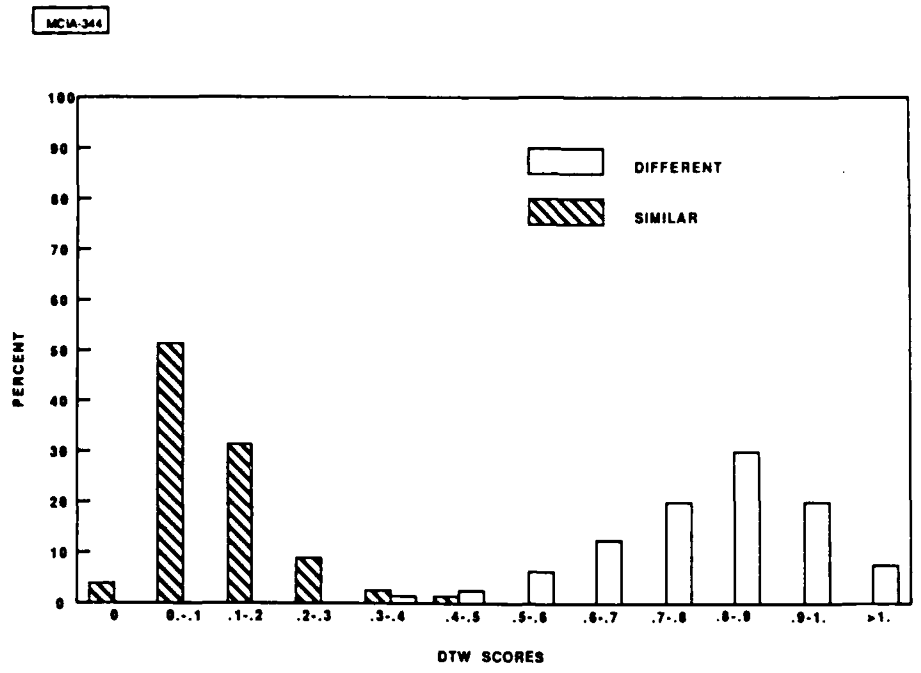


a. 2-Level Quantization

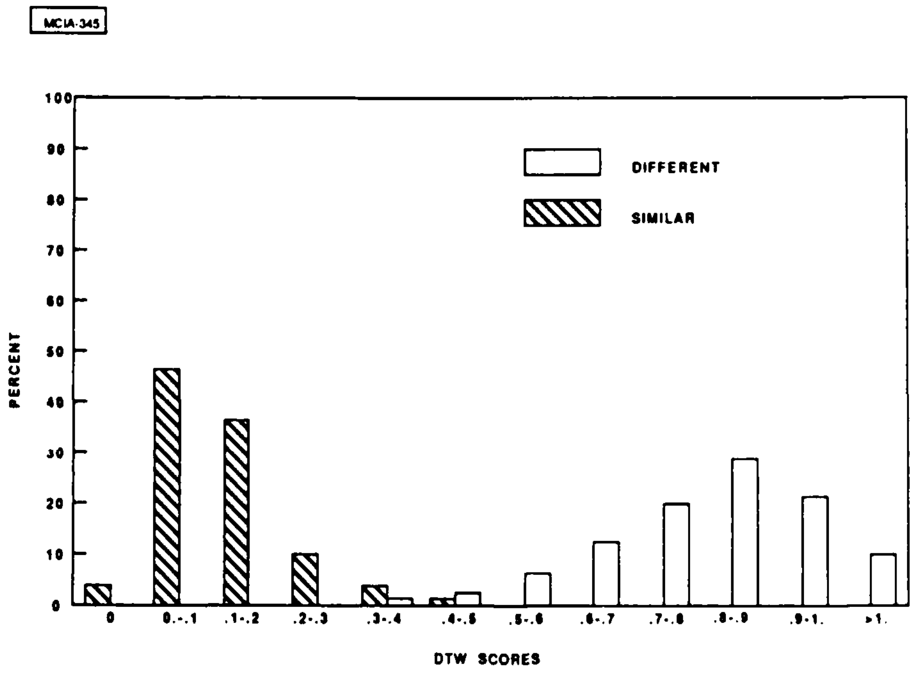


b. 3-Level Quantization

Figure 3.31. DTW Scores with Pre-Emphasis: $\mu = .95$, RNS1: {23,11,2}, RNS2: {17,13}



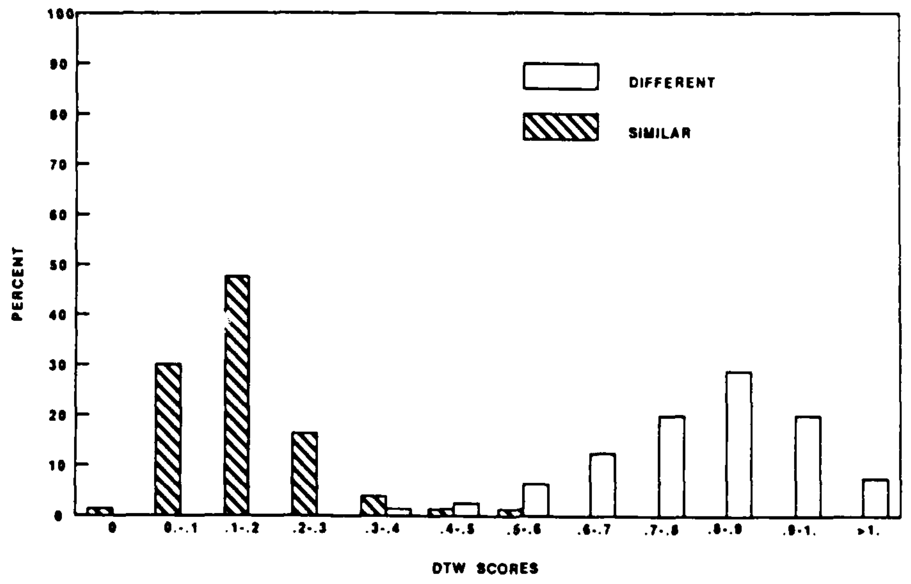
a. 2-Level Quantization



b. 3-Level Quantization

Figure 3.32. DTW Scores with Pre-Emphasis: $\mu = .95$, RNS1: {23,11,2}, RNS2: {13,11}

MCIA-346



a. 2-Level Quantization

MCIA-347

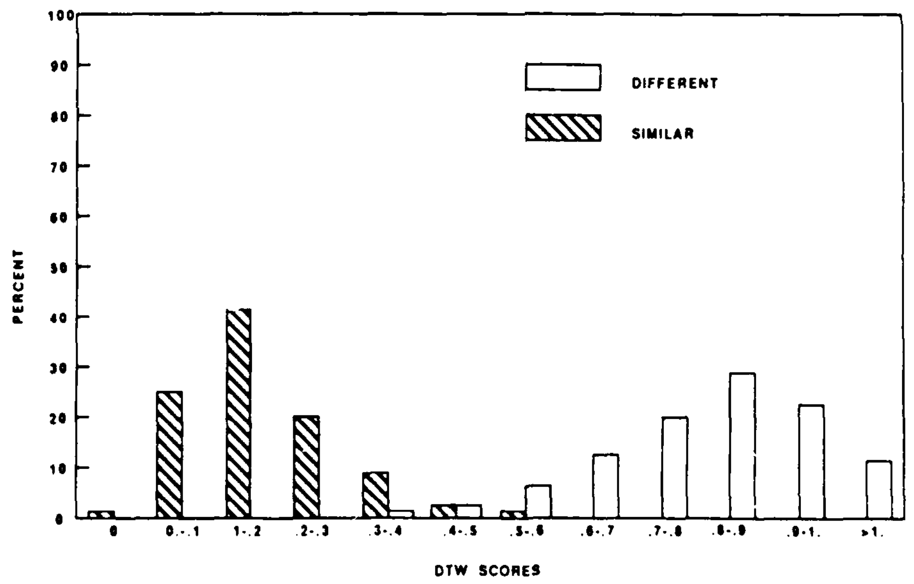
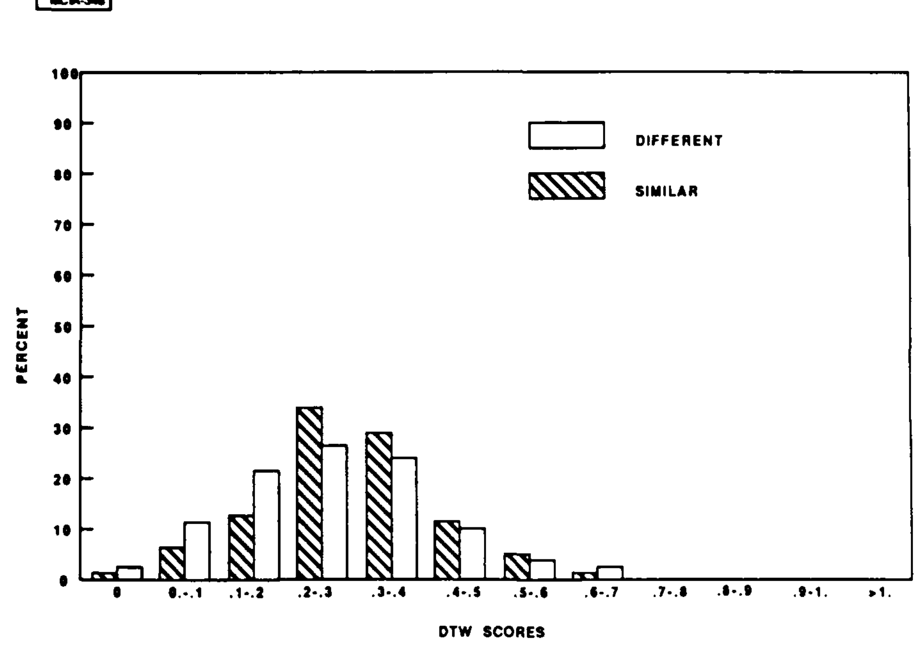


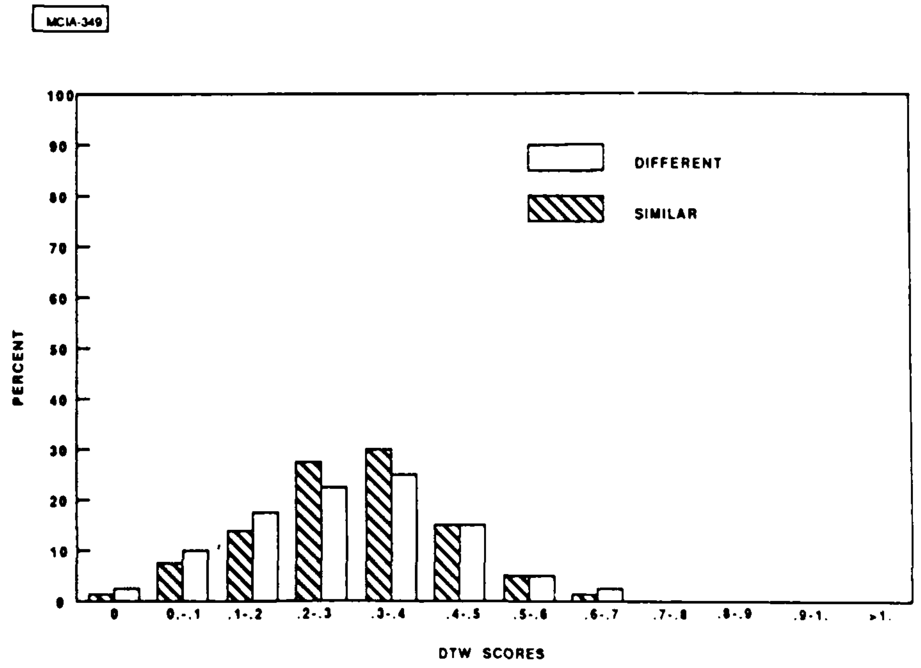
Figure 3.33b. HISTOGRAM OF DTW SCORES WITH PRE-EMPHASIS: MU = .95

b. 3-Level Quantization

Figure 3.33. DTW Scores with Pre-Emphasis: μ = .95, RNS1: {23,11,2}, RNS2: {11,7}



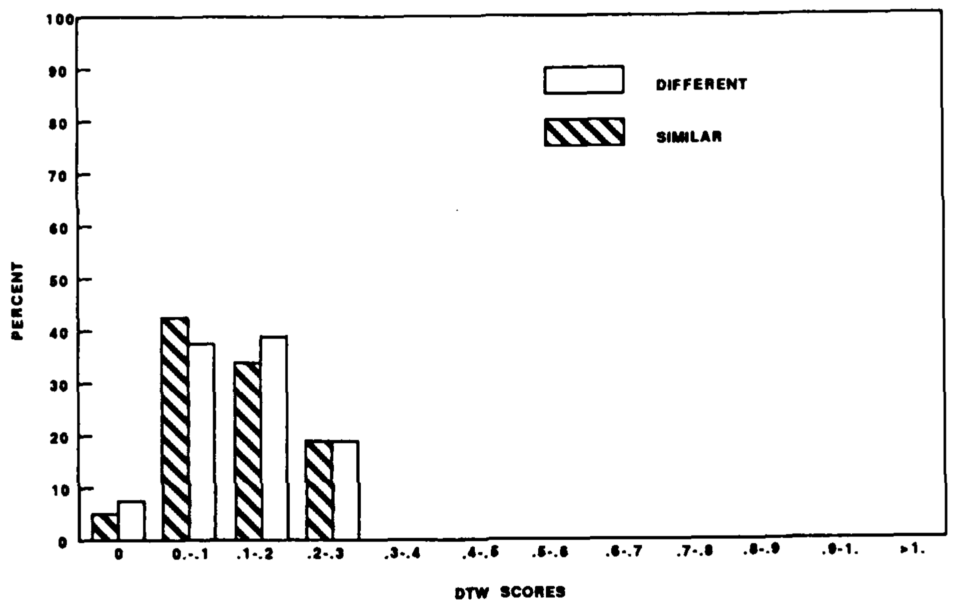
a. 2-Level Quantization



b. 3-Level Quantization

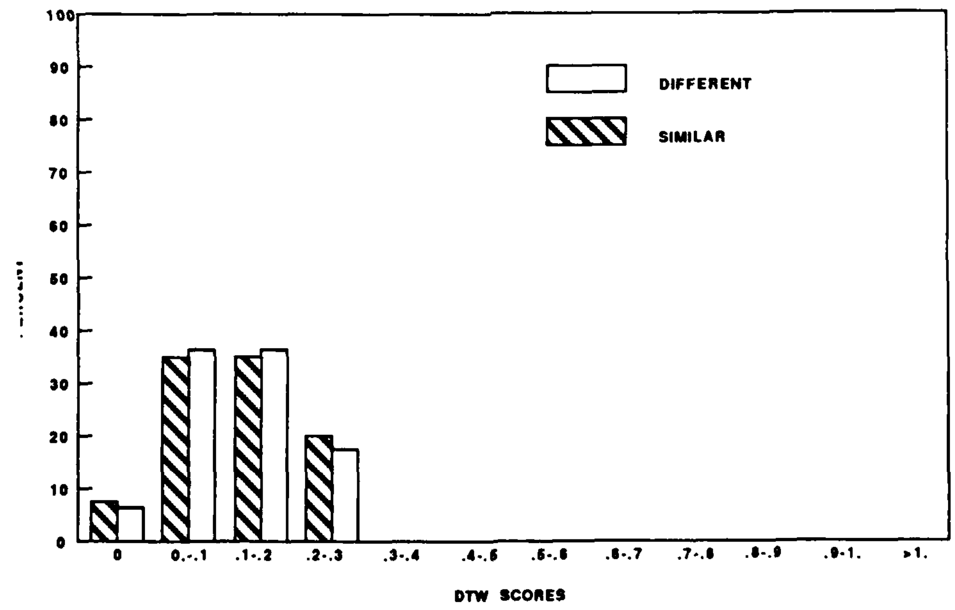
Figure 3.34. DTW Scores with Pre-Emphasis: $\mu = .95$, RNS1: $\{23,11,2\}$, RNS2: $\{7,5\}$

MCIA-350



a. 2-Level Quantization

MCIA-351



b. 3-Level Quantization

Figure 3.35. DTW Scores with Pre-Emphasis: $\mu = .95$, RNS1: {23,11,2}, RNS2: {5,3}

3.9 EXPERIMENTS WITH A VOCABULARY OF SIMILAR SOUNDING WORDS

Recognition results obtained using the 37-word database described in section 3.1 have been good, reaffirming the findings of earlier simulations based on an 11-word database. We expect these results to be valid for much larger databases, but have been unable to confirm this because of the large amount of computer time required to run the RNS word-recognition simulations.

We believe that matches of different productions of the same word will continue to exhibit low DTW scores independent of the size of the database used for building the reference library. The real impact of an expanded database arises from the increased likelihood that there will be other words different from the test input but close enough to it in sound to produce low DTW scores when matched with the test. This has not been tested by the 37-word database, since the 26 communication codewords were selected presumably for their properties of distinctness.

In order to study the performance of our RNS implementation of a DTW-based word-recognition system when the database vocabulary contains similar sounding words, we have created another database, the rhymes database, containing 27 words. Five productions of each word were recorded by speaker A to form a database for reference library construction. A sixth production was recorded to serve as test input. The number of frames produced by the utterance segmentations are shown for all words in table 3.7.

Table 3.7

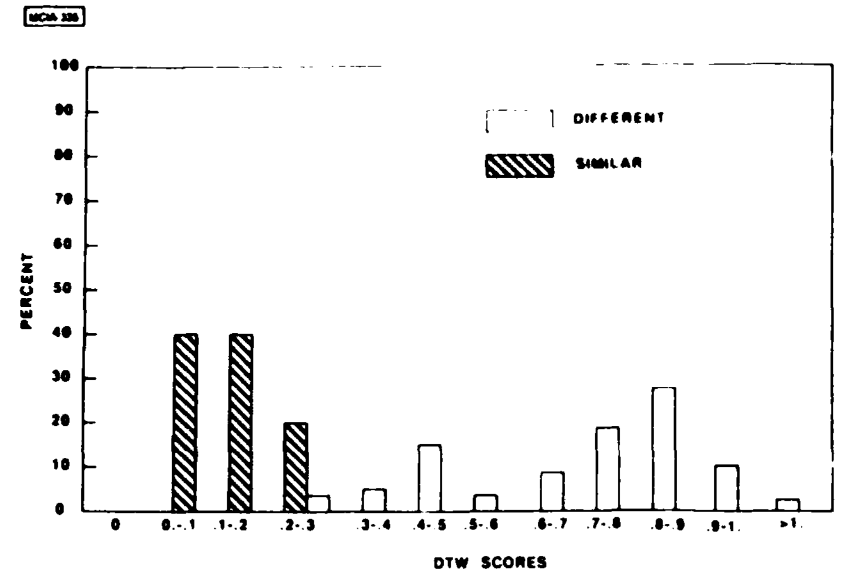
Number of Segments Produced for Five Reference Productions and One Testset Production for Rhymes Database Segmentations

Word	Prodn1	Prodn2	Prodn3	Prodn4	Prodn5	Test
mack	48	41	46	44	45	49
man	61	67	73	65	67	59
mat	54	48	47	46	46	55
mech	57	46	46	44	44	40
men	56	57	54	58	54	56
met	44	44	40	42	43	44
mick	47	44	45	47	47	50
min	55	53	55	54	54	51
mitt	55	51	53	47	48	53
pack	48	48	46	46	36	49
pan	52	50	56	55	57	50
pat	51	53	51	46	44	50
peck	42	36	40	37	40	42
pen	57	54	52	52	53	53
pet	47	44	46	46	42	38
pick	42	42	44	42	42	39
pin	48	50	50	53	52	44
pit	47	46	49	47	39	45
sack	55	39	52	63	58	68
san	70	56	72	63	75	70
sat	44	49	60	60	62	58
sec	72	55	64	49	72	56
sen	63	61	69	70	72	64
set	58	53	54	41	43	54
sick	61	62	62	52	54	61
sin	66	48	67	62	64	57
sit	49	51	52	49	53	53

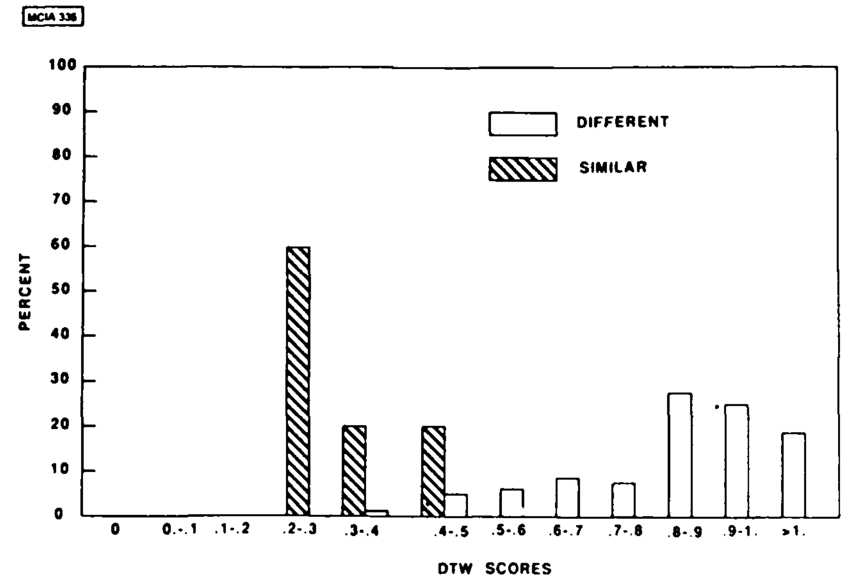
Each word of the rhymes database has six near neighbors, e.g., the word "mack" is a neighbor of "man" and "mat," differing only in the terminal consonant; of "mech" and "mick," differing only in vowel sound; and of "pack" and "sack," differing only in initial consonant.

Simulations were run with a library constructed from this database, with PORDER = 12, 10-bit input quantization, and pre-emphasis of the speech samples. The histograms of DTW scores for matches of similar words and matches of different words are shown in figure 3.36a (RNS {23,11,2}) and figure 3.36b (RNS {13,7,5}). The scores for matches of similar words are somewhat higher and more spread out than before (cf. figure 3.2). The scores for matches of different words are lower, as expected. Their histogram is shifted to the left (relative to figure 3.2). Separation between histograms for matches of similar words and those for matches of different words has deteriorated considerably, but still is sufficient to hope for reasonably good performance of a word-recognition system based on larger vocabularies.

For the simulation employing the RNS {23,11,2}, a single recognition error resulted. The test input "met" was confused with "mat," both reference words returning the same normalized path score. For the simulation using the RNS {13,7,5} no recognition errors occurred. The histograms of figure 3.36 show comparable performance for each RNS. For the RNS {23,11,2}, 71% of the correct matches have lower scores than 96% of the incorrect matches; for the RNS {13,7,5}, 70% of the correct matches have scores lower than 97% of the incorrect matches; for the RNS {23,11,2}, 96% of the correct matches have scores lower than 89% of the incorrect matches; for the RNS {13,7,5}, 97% of the correct matches have scores lower than 84% of the incorrect matches.



a. RNS {23,11,2}



b. RNS {13,7,5}

Figure 3.36. DTW Scores with Pre-Emphasis: μ = .95, 10 Bit Inputs, Rhymes Vocabulary

SECTION 4

RNS-SYSTOLIC IMPLEMENTATION

Two systolic processing arrays are described in this section, one to compute the correlation values of the individual frames of speech and one to perform DTW. Both utilize RNS. The linear array for correlation computation has been discussed in [1] but is included briefly for completeness. The elementary cell in the DTW array presented here uses the square of the Euclidean distance as a distortion function, and the quantization of the distortion values is done by mixed-radix conversion. The underlying systolic configuration was originally conceived for general dynamic programming [9].

To provide a very rough estimate of the upper limit of hardware complexity of the DTW array, layouts of the reduced logic expressions of the essential functions were examined using available CAD tools for simple programmable logic arrays. Projections of hardware complexity and throughput based on a careful state-of-the-art VLSI design are extrapolated from these results.

4.1 A SYSTOLIC AUTOCORRELATION VECTOR COMPUTER

Let $x(m)$, $m \geq 0$, be a set of uniformly sampled values of the speech signal. This sequence is divided into finite, connected portions that represent separate utterances; these utterances are compared, one by one, with every utterance in a library.

From a given test utterance we construct overlapping segments of M samples; the shift between consecutive segments is denoted A. Typical values of M and A are 180 and 80, respectively. The ℓ-th segment (or frame) is denoted $x^{(\ell)}(m)$, $0 \leq m \leq M - 1$. From each frame we define an autocorrelation vector $\underline{r}^{(\ell)} = \{r_P^{(\ell)}, \ldots, r_0^{(\ell)}\}$, where P is the order of the autoregressive model, according to the formula

$$r_n^{(\ell)} = \sum_{m=0}^{M-1-n} x_m^{(\ell)} x_{m+n}^{(\ell)}. \tag{18}$$

The order that we use is P = 12.

4.1.1 Systolic Array

Since frames overlap, multiple correlators are required so that the correlation vectors can be computed quickly. With M = 180 and A = 80, at most three frames can overlap. Three correlators are needed. This is illustrated in figure 4.1, where the input switch selects samples from the appropriate frames.

IA-71,960

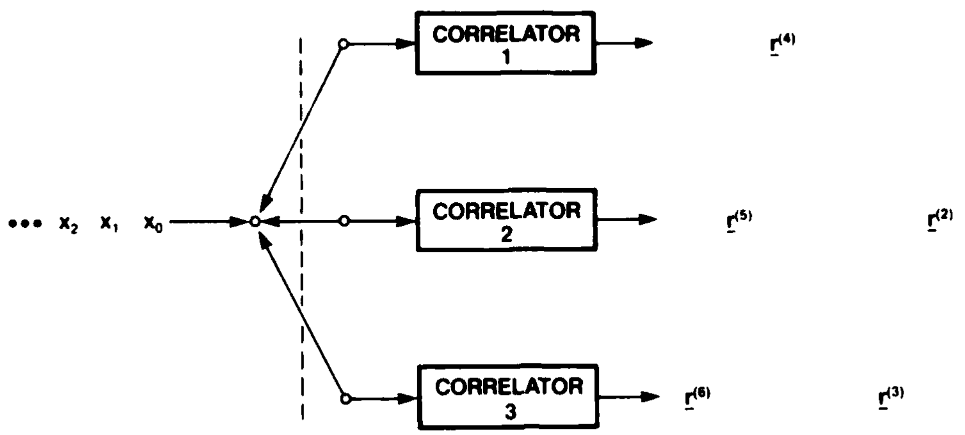


Figure 4.1. Autocorrelation Computer

Each correlator can be implemented with the linear systolic array shown in fiqure 4.2, where all P + 1 cells are identical. Two copies of the input samples for one segment enter the array, one at each end, with zeros interleaved. Computation beqins when both copies of x_0 appear at the inputs of cell 0. The data then proceeds, one cell at a time, through the array. At each time instant each cell forms the product of its two inputs and adds it to the contents of an accumulator. After all input samples in the segment have been used, each cell contains its corresponding coefficient.

IA-71,961

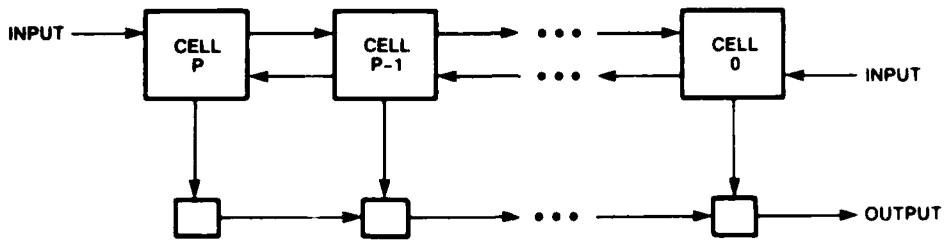
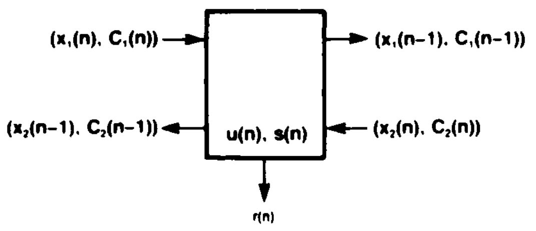


Figure 4.2. Systolic Autocorrelation Computer

The defining equations of the elementary cell are shown in figure 4.3. If A is given a value of M/3, the two segments to be computed consecutively in one correlator are not separated by zeros, and something else must be done to prevent results corrésponding to different segments from mixing in the same cell. For this purpose two control bits can be attached to the incoming data; one accompanies x_0 of the right input, and the other is attached to x_{M-1} of the left input. Control bit C_2 (figure 4.3) causes a cell to output the contents of its accumulator and start a new computation, and control bit C_1 instructs the cell to ignore further inputs. The basic operation of the array is illustrated in figures 4.4 and 4.5; the interplay of the control bits is shown in figure 4.6.

IA-71,959



$$u(n+1) = \begin{cases} 1 \text{ IF } n+1 < 0 \\ u_n \text{ IF } C_1(n) = 0 \\ u_n' \text{ IF } C_1(n) = 1 \text{ OR } C_2(n) = 1 \end{cases}$$

$$s(n+1) = \begin{cases} 0 & \text{IF } n+1 < 0 \\ s(n) + x_1(n)\,x_2(n) & \text{IF } u(n) = C_2(n) = 0 \\ s(n) & \text{IF } u(n) = 1,\ C_2(n) = 0 \\ x_1(n)\,x_2(n) & \text{IF } C_2(n) = 1 \end{cases}$$

$$r(n+1) = \begin{cases} s(n+1) \text{ IF } C_2(n) = 1 \\ r(n) \text{ IF } C_2(n) = 0 \end{cases}$$

Figure 4.3. Processing Element in Autocorrelation Array

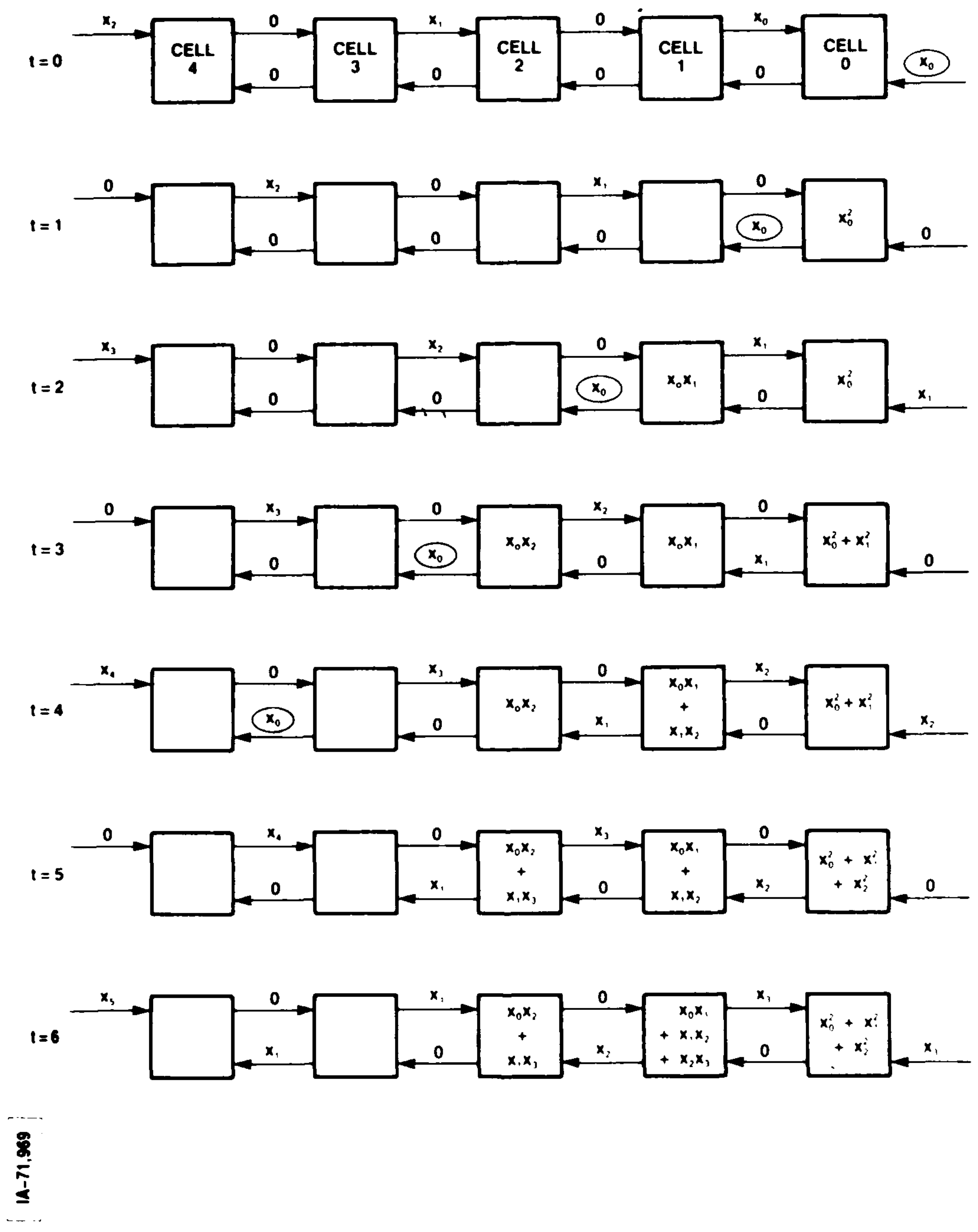


Figure 4.4. Autocorrelation Computation

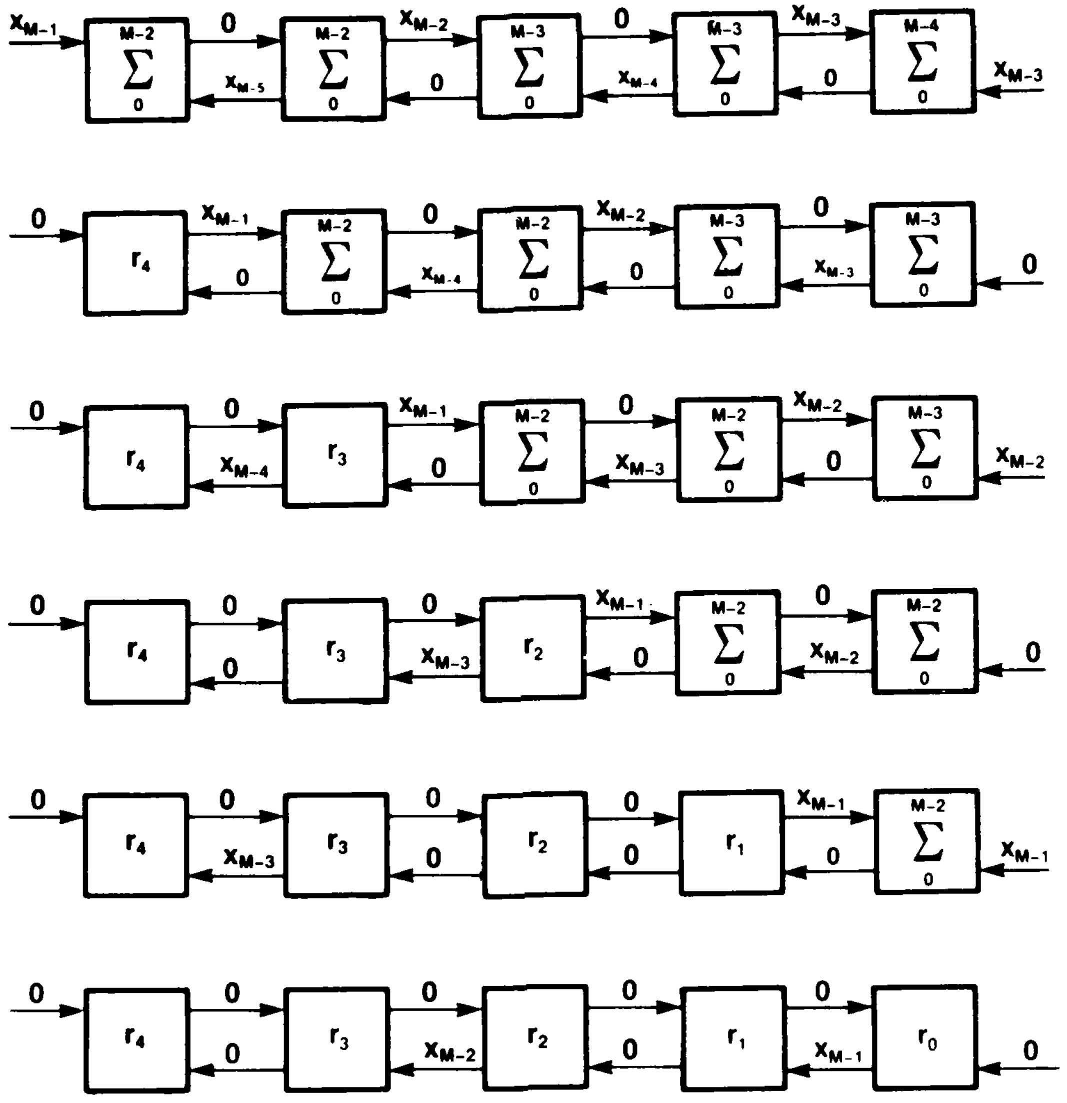
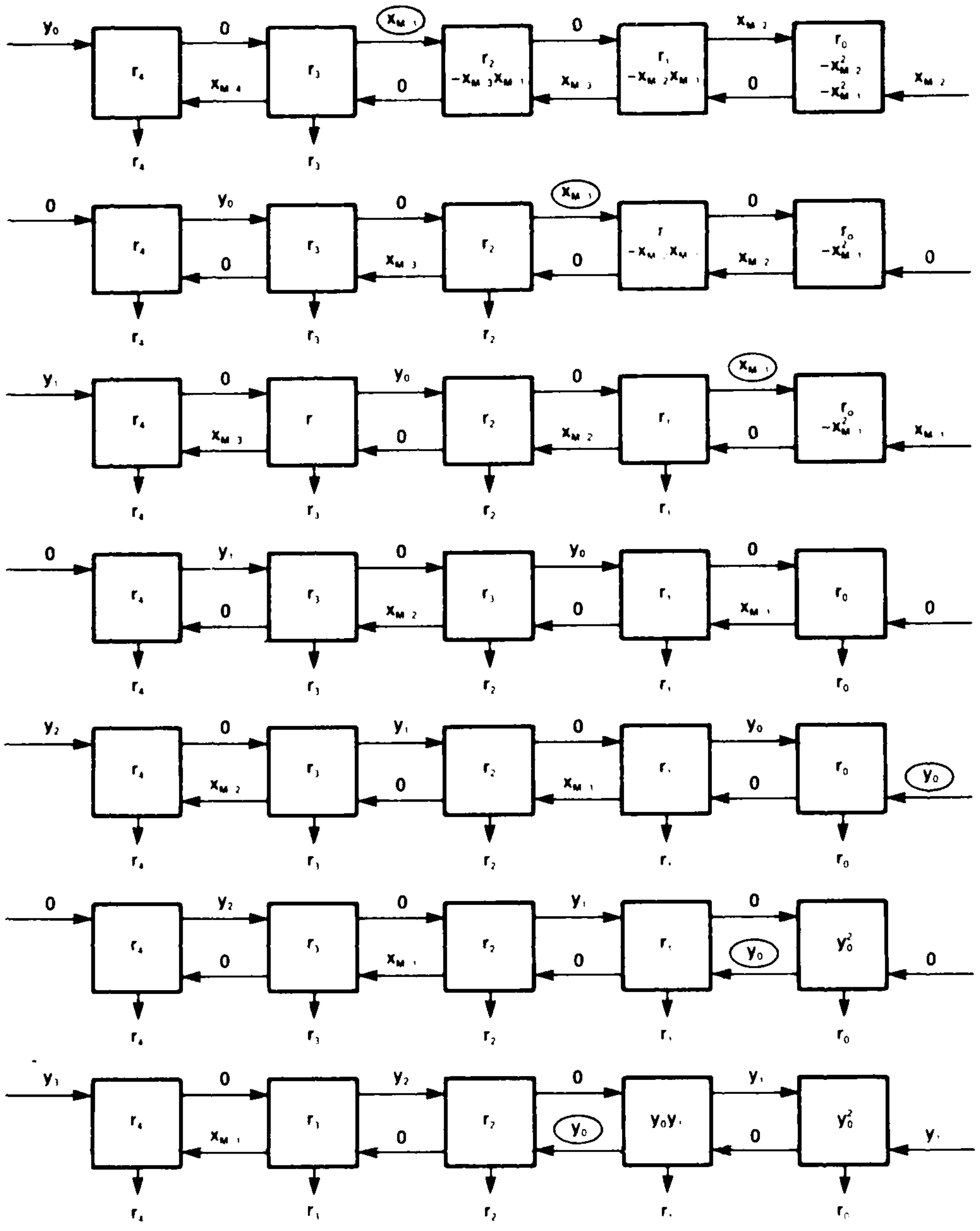


Figure 4.5. Autocorrelation Computation: Final Steps



IA-71,970

Figure 4.6. Autocorrelation Computation:
Two Consecutive Input Segments

4.1.2 RNS Hardware Implementation

If the set $p_1, p_2, \ldots, p_\ell$ is used for the moduli of the RNS, then ℓ versions of each correlator are required, each one operating with arithmetic modulo p_i. One RNS correlator is shown in figure 4.7. (The small squares are time delays used to impart the appropriate time lag to the data input from the right.) A more detailed design of a modulo P cell appears in figure 4.8. This diagram illustrates a modification of a cell previously designed [10] for use in a transversal filter.

4.2 NORMALIZATION

The correlation vectors are normalized before they are processed by the DTW algorithm.

The normalized correlation vector $\underline{r} = (r_P, \ldots, r_0)$ is obtained by dividing each component by r_0. Since $0 \leq |r_n| \leq |r_0|$ for each $n = 0, 1, \ldots, P$, all the normalized components lie between plus one and minus one. The normalized values are then scaled and quantized to an integer between 0 and S, where S is the scale factor. Then they must be encoded in the RNS used in the DTW.

Division by a variable number is too complicated when the numbers are represented by their residues; we must exit from RNS for general division. A weighted (binary) number representation of all the r_n can be obtained by mixed-radix conversion so that the division by r_0 can be performed. This division occurs only once for each test segment processed, while each normalized vector is used many times in the DTW operation.

IA-73,211

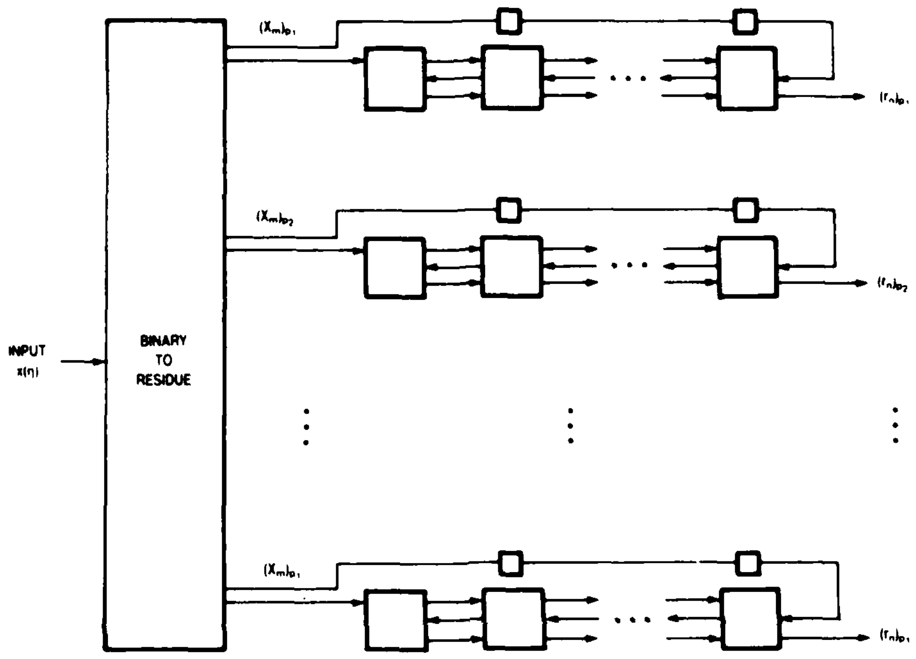
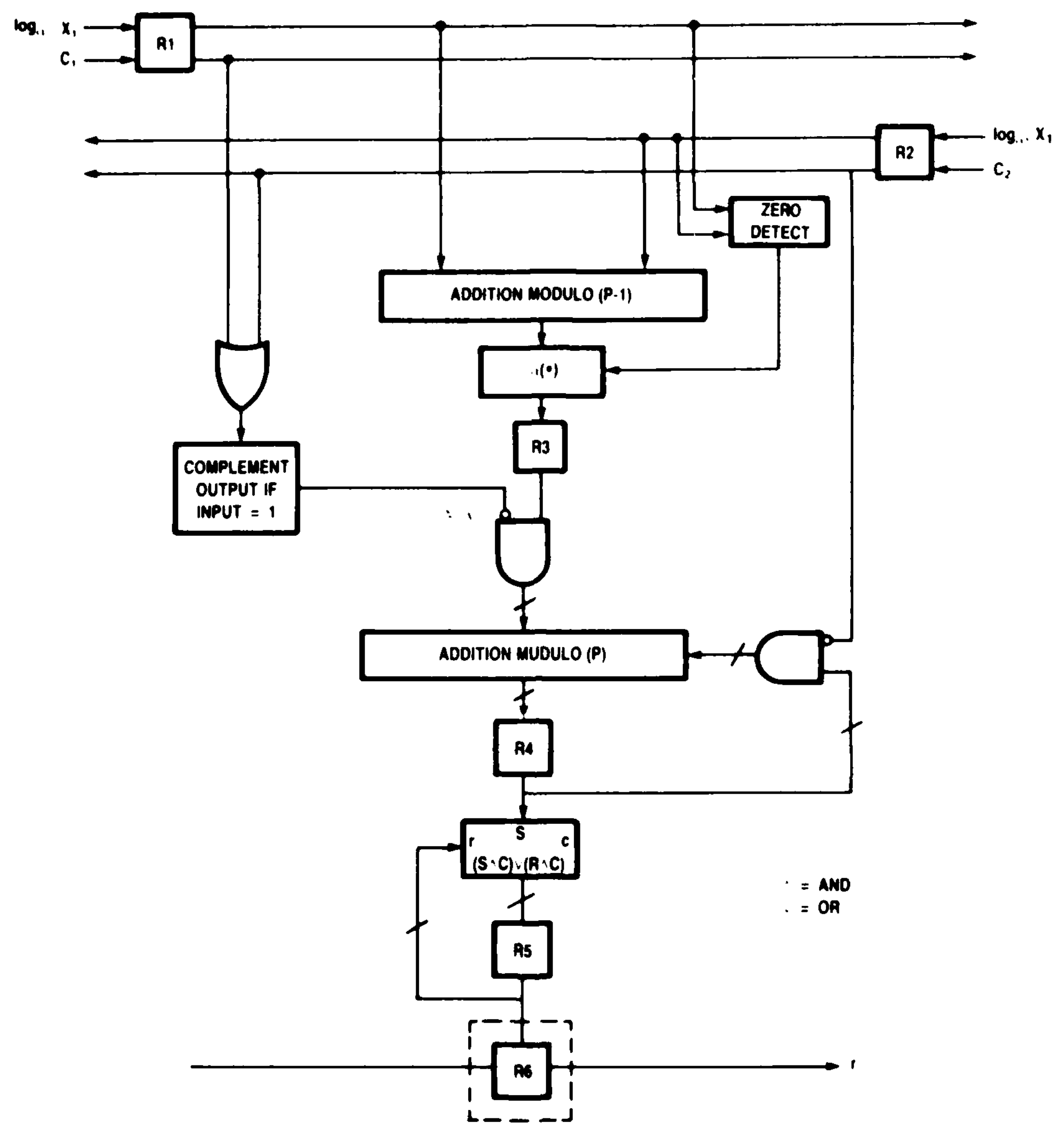


Figure 4.7. Systolic RNS Autocorrelation Computer



IA-72,966

Figure 4.8. Systolic Correlator Cell Mod P

A block diagram of the operation is shown in figure 4.9. The correlation values are given in residue form with respect to the set of moduli used for their computation; each one of them is accompanied by a control bit which has a value of "one" for r_0 and "zero" for r_1, r_2, ..., r_p. The binary representations of the coefficients are obtained from a residue-to-binary converter. The r_0-divider recognizes r_0 by observing the control bit and is latched; the other coefficients in the same vector are divided by r_0. The value of r_0 is updated every time a new set of coefficients appears.

From the weighted-number representation of the normalized coefficients the new residues can be readily computed by a table lookup, which observes only the higher-order digits; for example, if S = 15, only 4 bits are required to determine all the residues in the new RNS.

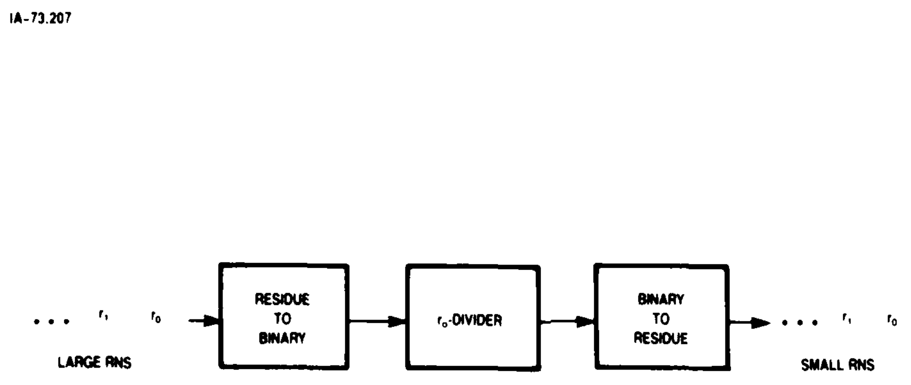


Figure 4.9. Normalization

IA-71,966

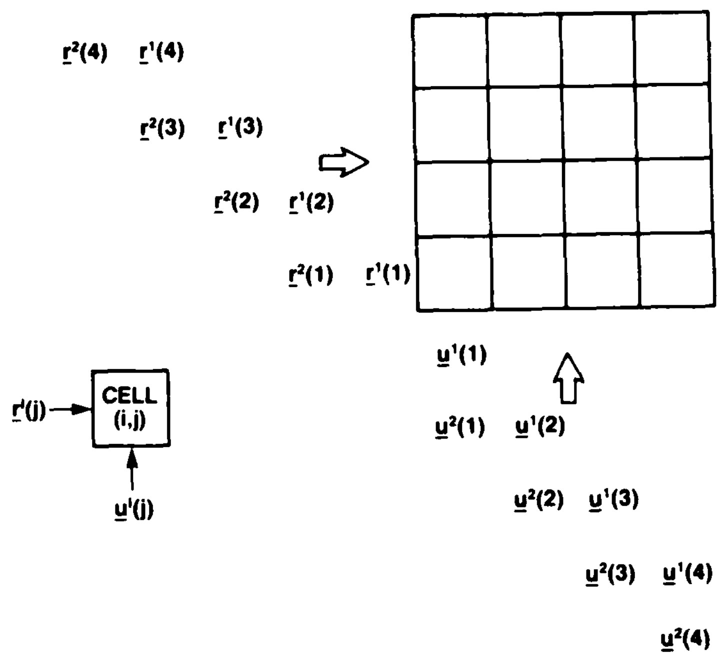


Figure 4.10. DTW Input Data Flow

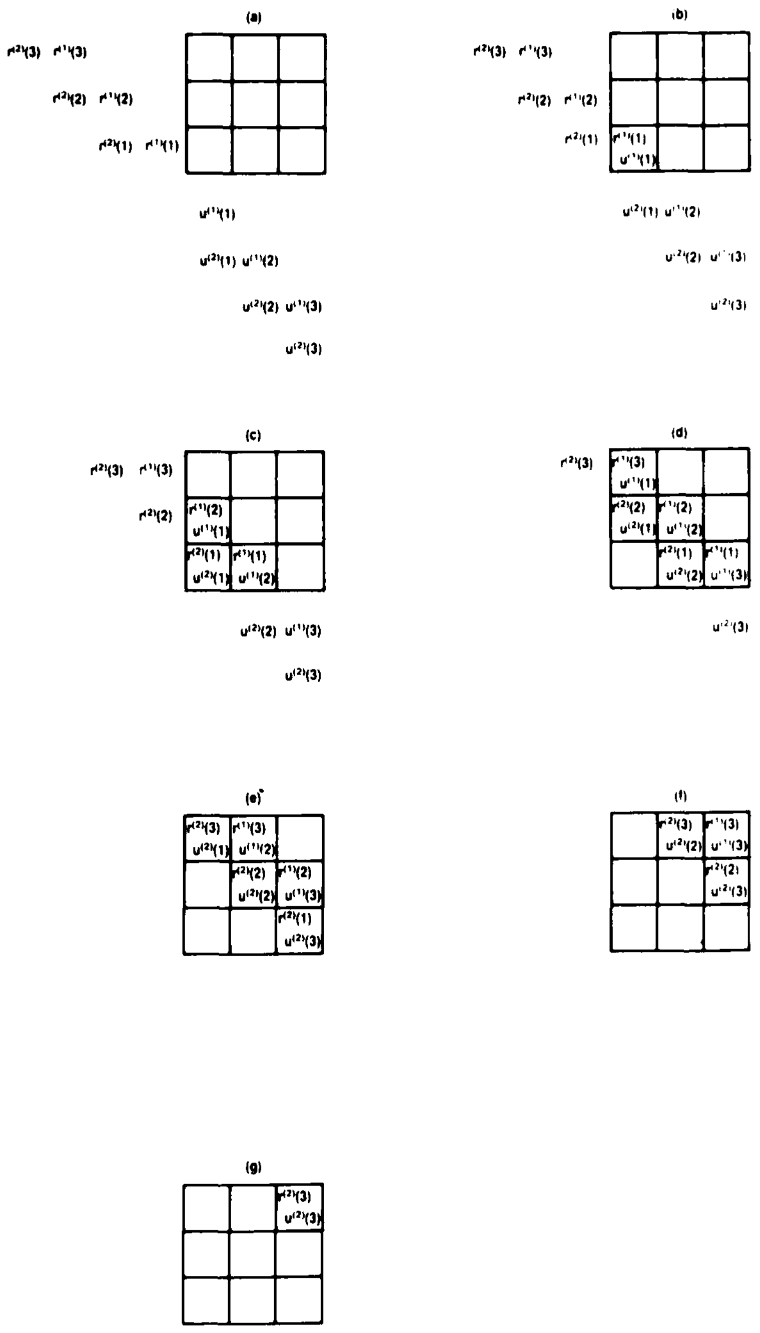


Figure 4.11. Systolic Computation of Local Distortion

4.3 SYSTOLIC ARRAY FOR DTW COMPUTATION

The DTW algorithm computes the cost of the shortest (lowest-cost) path through a two-dimensional array of distortion values determined by a pair of test and reference utterances.

Dynamic time-warping can be accomplished efficiently with a two-dimensional systolic array. The reference and test utterances, represented by their correlation vectors $\underline{r}(j) = (r_p(j), r_{p-1}(j), \ldots, r_0(j))$, $j = 1, 2, \ldots, n$, and $\underline{u}(i) = (u_p(i), u_{p-1}(i), \ldots, u_0(i))$, $i = 1, 2, \ldots, m$, respectively, enter the array as shown in figure 4.10. All data corresponding to one pair of utterances to be compared lie on one diagonal and bear the same superscript. As the data progress through the systolic array each diagonal retains its relative position with respect to the others so that all cells on a given diagonal operate on an utterance pair at any given time (see figure 4.11). Observe that cell (i,j) always receives the vectors $\underline{u}(i)$ and $\underline{r}(j)$ in an utterance pair.

Since the path computations for each pair of test and reference utterances will proceed as a wavefront making computations on successive diagonals, the deletion of previously used distortion values allows pipelining to the extent that distortion functions associated with a number of utterances equal to the number of diagonals can be present at any given time, with the path computations being pipelined along successive diagonals.

The computation of the shortest path is carried out iteratively. At each time instant each cell computes the shortest path from cell (1,1) to itself by observing the lowest costs of paths leading to the four cells that precede it (according to type 3 local path constraints), selecting the smallest one and then adding the local distortion, i.e., the cost of getting from that cell to itself.

At each instant each cell performs the following operations:

1. Computation of local distortion

$$d_{ij} = \sum_{n=0}^{p} (u_n(i) - r_n(j))^2 \tag{19}$$

2. Quantization of d_{ij} to d'_{ij} to reduce the numerical range requirement

3. Calculation of path costs

$$\begin{aligned} \hat{C}_{i,j} &= d'_{ij} + \min(C_{i-1,j-1}, C_{i-1,j-2}) \\ C_{i,j} &= \min(\hat{C}_{i,j}, d'_{ij} + \hat{C}_{i-1,j}). \end{aligned} \tag{20}$$

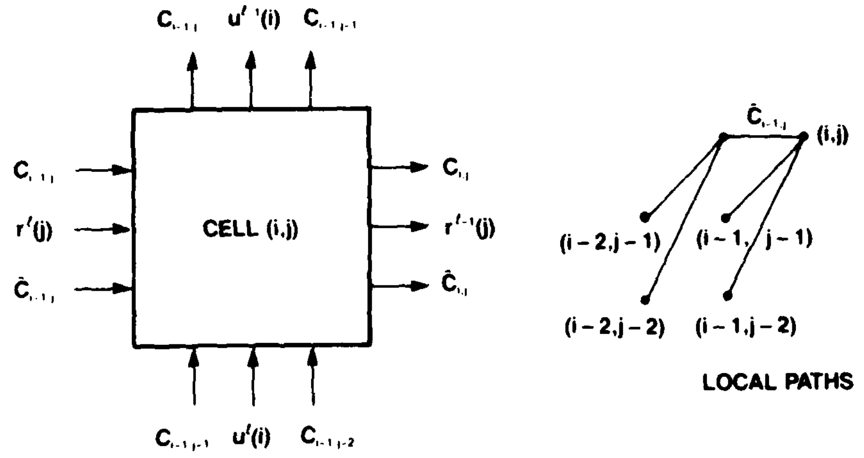
The results of step 3 are then passed along to neighboring cells for further processing.

For the computation in equation (20) to take place, the four path costs must be available at the input of cell (i,j) when the cells on its diagonal are ready to compute. This is clearly possible since the four path costs are on diagonals that have already completed computation. In the physical array all data move horizontally or vertically, and diagonal communication--e.g., the communication of $c_{i-1,j-1}$ from cell (i-1,j-1) to cell (i,j)--is done through cell (i,j-1). At each step on such a path, the data advance one diagonal. All data being operated on, or computed, corresponding to one pair of utterances, lie on the same diagonal.

From this observation, it follows that the distortion values on a given diagonal need not be available until the computational wave-front has reached that diagonal and this can be managed by the pipelined scheme already discussed for the distortion computations. Hence, the DTW can be completely pipelined, with each diagonal handling one pair of utterances. From cell (m,n) the scores of the pairs of utterances compared will then emerge one-by-one, and the entire process is readily pipelined.

Figure 4.12 shows a typical cell in the DTW grid. The computation of c_{ij} is done in two steps so that actually only three quantities previously computed are fed to each cell ($\hat{c}_{i-1,j}$ is the minimum of $c_{i-2,j-1} + d_{i-1,j}$ and $c_{i-2,j-2} + d_{i-1,j}$). Also, $c_{i-1,j}$ is not used by cell (i,j) but is passed along from cell (i-1,j) to cell (i,j+1). Similarly, $c_{i-1,j-1}$ is passed to the cell above, after it has been used by cell (i,j). Finally, $\hat{c}_{ij}$ is computed and passed to cell (i+1,j). Thus, in addition to inputs $\underline{r}(j)$ and $\underline{u}(j)$, which get passed along after d_{ij} is computed, each cell accepts four inputs and produces four outputs.

IA-72,968



1. $d_{ij} = \sum_{n=0}^{p} \left(r_n^{\ell}(j) - u_n^{\ell}(i)\right)^2$
2. $\hat{C}_{i,j} = d_{ij}' + \min(C_{i-1,j-1}, C_{i-1,j-2})$
3. $C_{i,j} = \min(\hat{C}_{i,j}, d_{ij}' + \hat{C}_{i-1,j})$

Figure 4.12. DTW Array Processing Element

4.3.1 Local Distortion

In our RNS implementation three primes are used for the moduli, and d_{ij} is computed independently in each of the three residue channels. A block diagram of a serial, local-distortion computer is shown in figure 4.13. It consists of a mod p subtractor-squarer, a mod p adder and a latch. The mod 23 and mod 11 subtractor-squarers and adders can be implemented with PLAs; the mod 2 hardware requires only two exclusive-OR gates, one for each of the two functions.

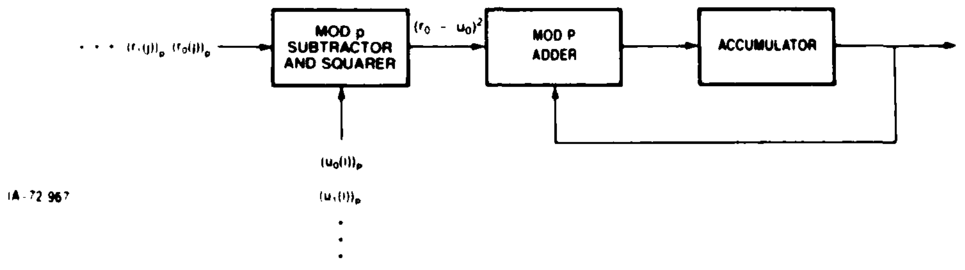
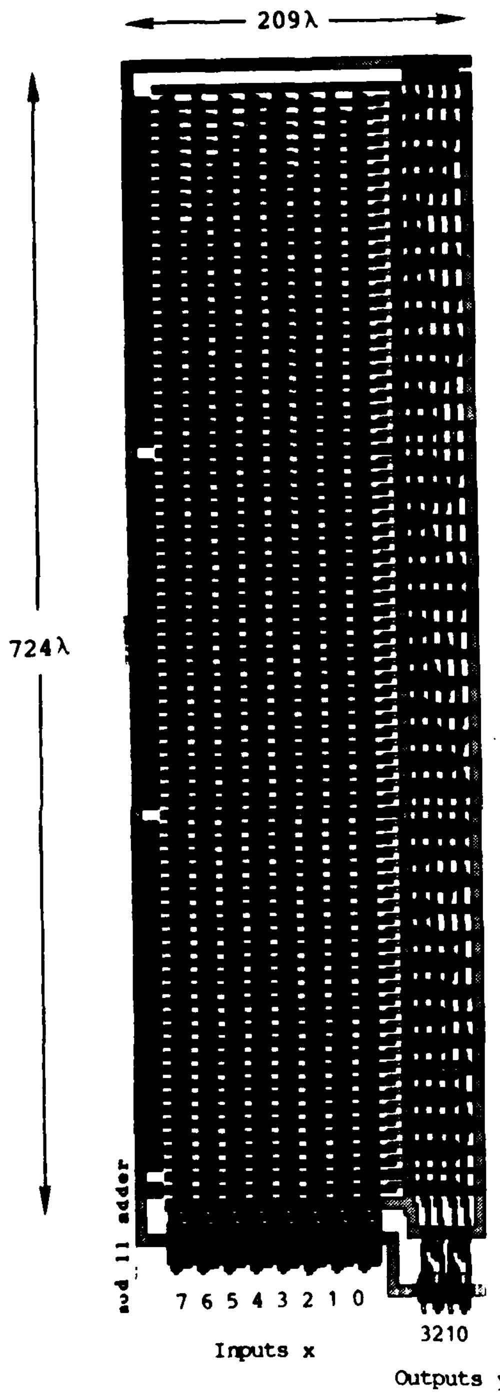


Figure 4.13. Local Distortion Computation Mod P

The U.C. Berkeley Boolean function reduction program ESPRESSO was used to simplify the two mod 23 and the two mod 11 functions. The reduced Boolean logic expressions are tabulated in the appendix. Computer-generated plots of the PLAs corresponding to the reduced mod 11 subtractor-squarer and adder are shown in figure 4.14. Both PLAs have eight (one-bit) inputs and four outputs. The subtractor-squarer has 78 product lines and the adder has 79 product lines, so the PLAs are of equal size. The dimensions of the mod 11 functions are shown in figure 4.14 as multiples of the feature size λ.



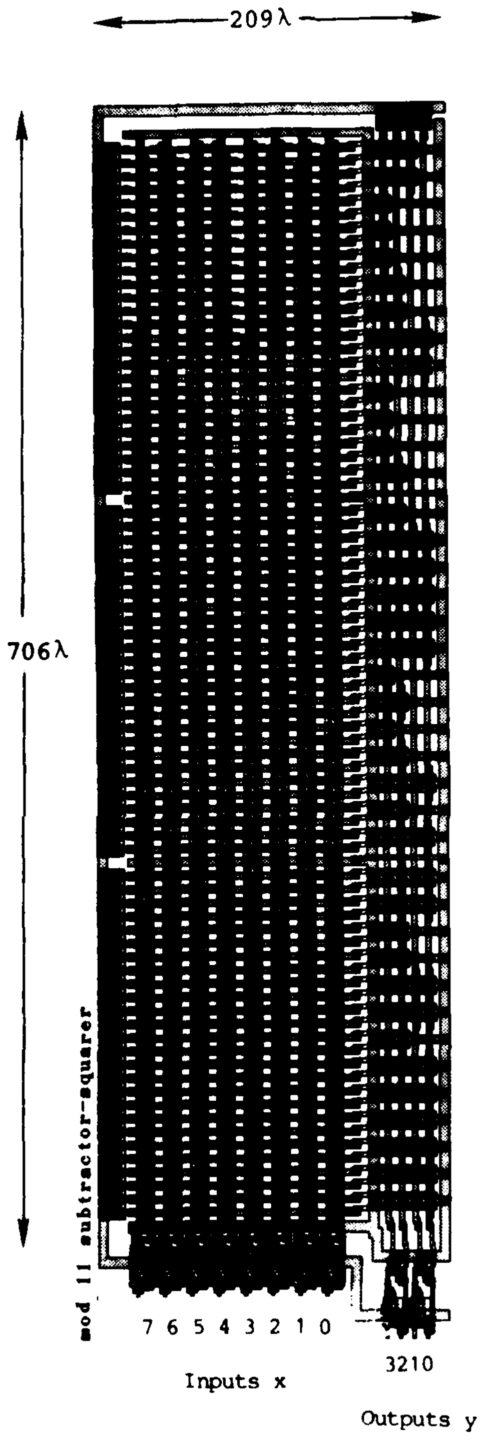


Figure 4.14. PLAs for Mod 11 Subtractor-Squarer and Adder

4.3.2 Mixed-Radix Quantizer

The two-level quantizer takes d_{ij} as input and lets the quantized value d'_{ij} be equal to 1 if $d_{ij} \geq 23$ and 0 if $d_{ij} \leq 22$. If d_{ij} has mixed-radix representation

$$d_{ij} = n_3 p_2 p_1 + n_2 p_1 + n_1 \tag{21}$$

then

$$d'_{ij} = \begin{cases} 0, & \text{if } n_2 = n_3 = 0 \\ 1, & \text{otherwise} \end{cases} \tag{22}$$

As explained in section 2, the condition $n_2 = n_3 = 0$ is equivalent to the pair of equations

$$\begin{aligned} d_{ij} &\equiv n_1 \bmod p_2 \\ d_{ij} &\equiv n_1 \bmod p_3. \end{aligned} \tag{23}$$

A circuit that performs the quantization is shown in figure 4.15. Since $d'_{ij} = 0$ or 1, it is identical to its residue in each class since $\{0,1\}$ is contained in every residue class.

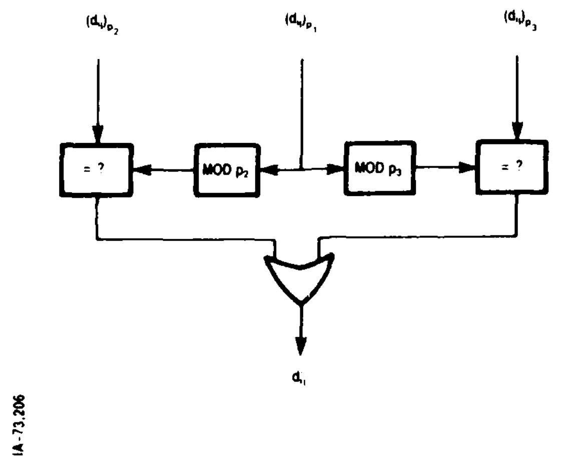


Figure 4.15. Two-Level Quantizer

For the RNS {23,11,2}, $p_1 = 23$; residues mod 23 must be reduced mod 11 and mod 2. The latter requires no further computation. The full quantizer can be implemented in the PLA shown in figure 4.16; it has 10 inputs (five, four and one bits for the mod 23, mod 11, and mod 2 residues, respectively), one output and 26 product lines. It implements a reduced Boolean expression obtained with Espresso. The espresso output for the full quantizer is tabulated in the appendix.

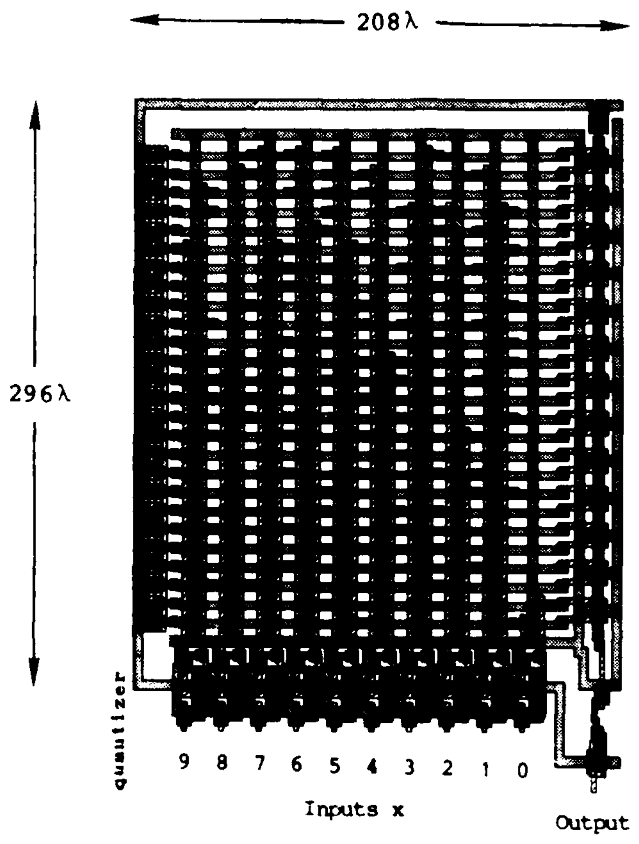
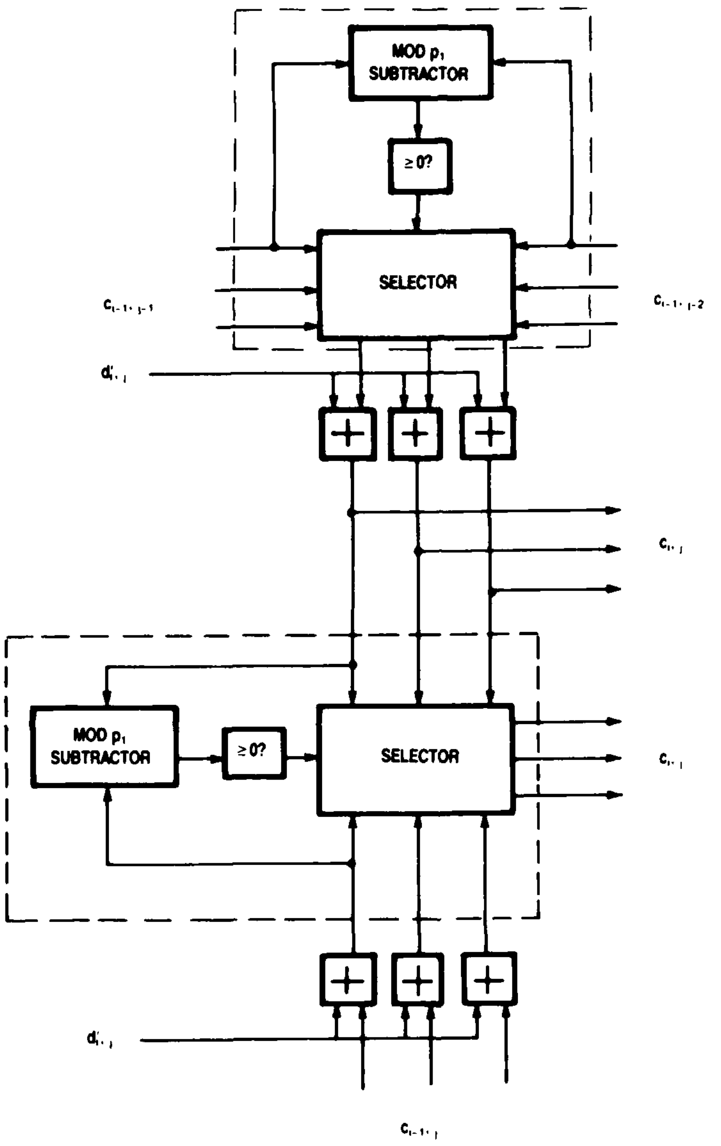


Figure 4.16. PLA For Quantizer

4.3.3 Path Computation

The determination of the minimums in equation (20) is performed only in the largest residue channel. Figure 4.17 depicts a circuit that computes $\hat{C}_{ij}$ and C_{ij} in an RNS of three residues.



IA-73,205-1

Figure 4.17. Path Computer

The dashed boxes enclose circuitry that computes minimums. The computation of each minimum requires a mod p_1 subtractor, a sign detector, and a selector. The selector is trivial and should occupy only a small area. The other two functions can be integrated and implemented in a 10-input, 1-output PLA. Again, Espresso was used to reduce the Boolean expression for this function; the reduced expression is given in the appendix. A PLA implementation is pictured in figure 4.18. It requires only 32 product lines. The adders used for d'_{ij} are quite simple since d'_{ij} is a one-bit number.

4.3.4 Packaging and Throughput

Each DTW cell requires the following functions:

One Mod 23 Subtractor-Squarer
One Mod 23 Adder
One Mod 11 Subtractor-Squarer
One Mod 11 Adder
One Quantizer
Two Minimum Computers

To obtain a crude estimate of the area of one DTW cell, PLAs implementing the above functions are grouped in figure 4.19. Table 4.1 summarizes the dimensions in terms of the feature size λ. The details of the required control circuitry and interconnections as well as a few gates required for the mod 2 operations have not been considered in this initial estimate. Assuming pessimistically

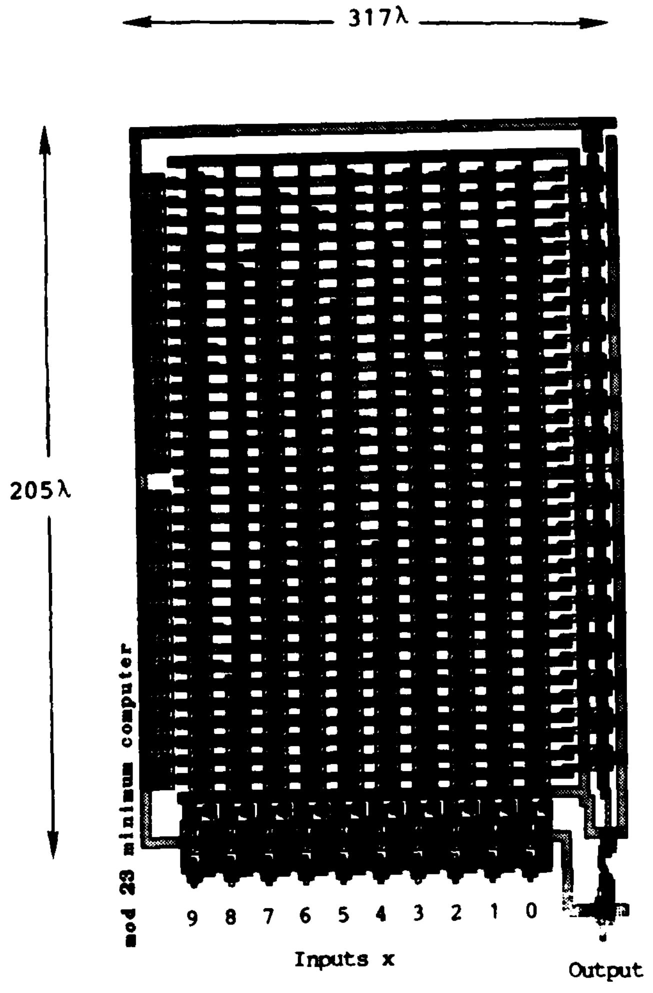


Figure 4.18. PLA for Mod 23 Minimum Computer

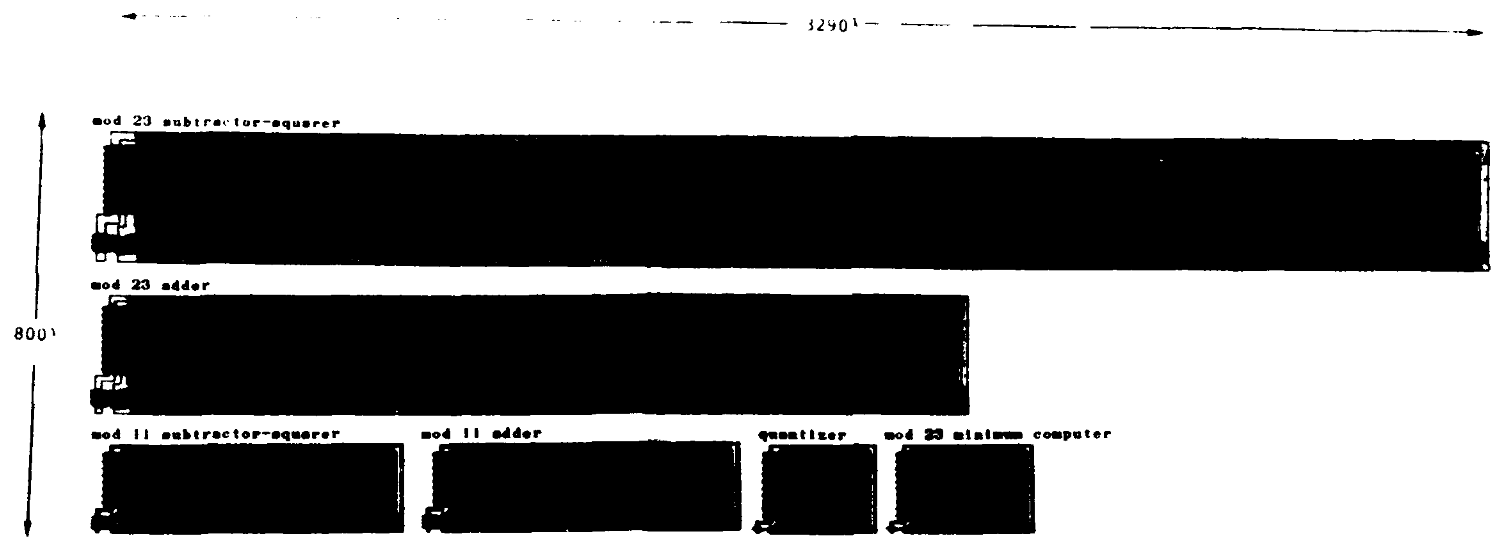


Figure 4.19. **PLAs for DTW Functions**

that this additional circuitry would represent 50% of the total hardware, we estimate that the chip area for one DTW cell is $3{,}200{,}000\lambda^2$ (an $800\lambda \times 4000\lambda$ rectangle) which in 4 μm nMOS technology represents 12.8 mm^2. It follows that in a 7 mm × 7 mm chip four cells might be integrated. This is an upper bound an area since, as shown in the appendix, a reduction of the area by a factor of more than ten is likely if random logic is used and, as discussed below, data would flow serially from cell to cell, so that some of the functions implemented here in PLAs could be simplified by using combinational logic and making use of the serial data flow. This would be especially important in the case of mod 23 functions because the corresponding PLAs are relatively large. A comparison of the area of a PLA and a custom logic design of a mod 23 adder is included in the appendix.

Table 4.1

PLA Dimensions for DTW Cell Functions

Mod 23 Subtractor-Squarer	$312\lambda \times 3290\lambda$
Mod 23 Adder	$276\lambda \times 2040\lambda$
Mod 11 Subtractor-Squarer	$209\lambda \times 706\lambda$
Mod 11 Adder	$209\lambda \times 724\lambda$
Quantizer	$208\lambda \times 296\lambda$
Mod 23 Minimum Computer	$205\lambda \times 317\lambda$

One advantage of systolic arrays is modularity, i.e., only one cell needs to be designed with multiple cells repeated in one IC. A problem with two-dimensional arrays is the I/O limitation. If an $N \times N$ array of cells is to be incorporated in one package, 4MN I/O leads are required. M is the number of leads connected to each side of each cell, assuming, of course, that more than one IC is interconnected to form the complete system.

Standard IC packages come with 20, 40, 84, and 132 pins; the latter two are pin grid arrays and the former two are dual in-line packages. Based on the pessimistic area estimate given above, at least four cells could be integrated in a 49 mm^2 chip. If random logic is used and the size of a cell is reduced so that 49 cells fit in a chip, M is reduced to 4.

The I/O limitation is important because the time required to transmit one bit between chips in 3 μm CMOS technology is about 100 ns. With a 12-pole model, 15 symbols must cross each side of each systolic cell: 13 correlation symbols and 2 path cost symbols. Since communication between residue channels is required, each cell in the array should contain the hardware for all residues. With 23, 11, and 2 as moduli each symbol requires 10 bits for its representation so 150 bits must cross in each cell period through 4 leads. This requires at least 3.75 μs, which is longer than all the other cell operations require. Hence, the time-warp rate of the systolic array is estimated as 266 kHz., as limited by the communication time. This rate can be increased at the expense of integrating fewer cells per chip.

4.4 CONCLUSION

Two systolic architectures were introduced in this section, one for computing autocorrelation coefficients and the other for performing DTW. Implementation in RNS was discussed, and trial PLA layouts were presented to estimate circuit complexity.

The computationally intensive part of the speech recognition system is the DTW, for which the corresponding systolic cell is simple when implemented in RNS. It is estimated that 49 cells could be custom-integrated into one chip, so for a 50 × 50 array about 50 ICs would be required. All chips would be identical, since each one performs computation in all the residue channels. Alternatively, wafer-scale integration could be used for the entire array.

The limiting factor in the speed of performance is the bit-serial transfer of information between two ICs, which, in 3 μm CMOS technology, can be done at a 10 MHz rate. Based on this rate, it is estimated that one new comparison between a test and a reference utterance can be done every 3.75 μs. Wafer-scale integration could also speed up the process.

SECTION 5

SUMMARY

Isolated word recognition is a computationally intensive processing function. The combination of residue number system computation and systolic array architectures offers practical simplification in the design of a special-purpose hardware processor. This report has described such an architecture; it uses short-time correlation analysis to form the spectral patterns, a distortion function employing squared Euclidean distance between the normalized correlation values of the test and reference utterance segments and a two-dimensional pipelined processor array to implement dynamic time-warping for pattern registration. With the exception of the normalization of the sample correlation values, all significant computations are carried out in a compact RNS for implementation in specially designed hardware of low complexity. This combination of techniques provides for a very high processing throughput in simple hardware that can be used for real-time word recognition with a large vocabulary.

This architecture has evolved from a previous attempt at implementation that used precomputed LPC analysis of the reference segments with the calculation of the Itakura-Saito distortion between the test and reference segments carried out in RNS. While the distortion calculation seemed well-suited to RNS, requirements for integer scaling of inverse correlation coefficients imposed an impractical size on the integer ring containing the computation. In the present method, we use the correlation values that determine a 12th-order LPC model, and, recognizing that the information requisite to predict the LPC model spectrum is contained entirely within these samples, we employ a squared Euclidean distance computation that reduces the size of the integer ring. We have been able to reduce the required computational range from about 30 bits for the Itakura-Saito distortion to about 9 bits for the squared Euclidean distortion without experiencing any significant loss in discrimination ability.

An underlying premise for performing the DTW computation in RNS, since it requires decisions based on comparison of the magnitudes of cumulative local-distortion differences, is that these differences are small enough to be contained within the range of a single modulus. The local decisions can then be made within a finite field while the resulting least-cost path metric is accumulated in the full RNS. To contain the range of the local distortion differences, we quantized the distortion values to a smaller range. In our architecture, this is done entirely in RNS by a partial mixed-radix conversion that establishes natural quantization boundaries and is simple to implement with Boolean logic.

We have described the processing algorithm, detailing the important steps; we have presented the results of extensive simulations using selected system parameters; we have also described the design concept and operation of a two-dimensional pipelined array that carries out both the local distortion computations and the DTW path-metric computations in an RNS of moderate range. We made a very rough estimate of the silicon area used and throughput attained from a hardware implementation in programmable logic arrays (PLAs) and exhibited some trial layouts for the reduced Boolean logic functions corresponding to functional components of the DTW array.

While we would not advocate implementation in PLAs because of the inefficient use of area, this artificial design exercise enabled us to upper-bound the complexity of the processor. Our conclusion is that the combination of RNS arithmetic calculating the squared Euclidean distance and quantized shortest-path search, when implemented in a two-dimensional pipelined array provides an architecture that is simple and practical, even if naively designed with PLAs. The careful design of these functions using state-of-the-art custom logic tools could provide an even simpler hardware implementation. We conclude that the architecture described is a leading candidate for VLSI implementation as a special-purpose hardware processor, a task that should be undertaken in a follow-on effort.

LIST OF REFERENCES

1. Bequillard, A.L., Carhoun, D.O., Eastman, W.L., "Advanced Architectures for Digital Signal Processors: Final Technical Report for Fiscal Year 1984," MTR9647, The MITRE Corporation, Bedford, Mass., April 1985.

2. Sakoe, H. and Chiba. S., "Dynamic Programming Algorithm Optimization for Spoken Word Recognition," IEEE Trans. Acoustics, Speech and Signal Proc., ASSP-26 (1978), pp. 43-49.

3. Myers, C., Rabiner, L.R., and Rosenberg, A.E., "Performance Tradeoffs in Dynamic Time-Warping Algorithms for Isolated Word Recognition," IEEE Trans. Acoustics, Speech and Signal Proc., ASSP-28 (1980), pp. 623-635.

4. Bellman, R., Dynamic Programming, Princeton University Press, 1957.

5. Moore, E.F., "The Shortest Path Through a Maze," Proc. of an Int. Symp. on the Theory of Switching, vol. II, Harvard University Press (Cambridge, 1959), pp. 258-292.

6. Dantzig, G.B., "Discrete-Variable Extremum Problems," Operations Research 5 (1957), pp. 266-270.

7. Bellman, R., "On a Routing Problem," Quarterly of Applied Mathematics, XIV (1958), pp. 87-90.

8. Knuth, D.E., The Art of Computer Programming: Volume 2/Seminumerical Algorithms, Addison-Wesley, Reading, MA; 1969.

9. Guibas, L.J., Kung, H.T., and Thompson, C.D., "Direct VLSI Implementation of Combinatorial Algorithms," Proc. Conf. Very Large Scale Integration: Architecture, Design, Fabrication, California Institute of Technology, January 1979, pp. 226-234.

10. Johnson, B.L., Vaccaro, J.J., Cosentino, R.J., "A VLSI-Based RNS-Systolic FIR Filter," MTR-9898 (to be published), The MITRE Corporation, Bedford, MA.

APPENDIX

REDUCTION OF THE BOOLEAN EXPRESSIONS FOR DTW CELL FUNCTIONS

The (modified) output of the U.C. Berkeley Boolean function reduction computer program ESPRESSO for the six functions discussed in section 4 is listed in tables A.1 through A.6.

The input variables are called $x_0, x_1, \ldots, x_{n-1}$ and the output variables $y_0, y_1, \ldots, y_{m-1}$. Consider the listing for the mod 11 subtractor squarer. The first three lines indicate that there are eight input and four output variables and 78 product terms. The rest of the printout consists of eight input and four output columns. Each eight-bit input word represents a product term in a Boolean sum-of-products expression. In each of the eight positions, a 1 means that the corresponding variable appears in the product term uncomplemented, a 0 means that it appears complemented and a - means that it is absent. For example, a product term listed as -1010000 stands for $x_1\bar{x}_2x_3\bar{x}_4\bar{x}_5\bar{x}_6\bar{x}_7$.

The reduced sum-of-product expression corresponding to the output variable y_i contains the product terms from the rows where a 1 appears under the column for y_i. For example, the first few terms for y_0 for the mod 11 subtractor-squarer are

$$y_0 = \bar{x}_0\bar{x}_1\bar{x}_2x_3x_5\bar{x}_6\bar{x}_7 + x_1\bar{x}_2\bar{x}_3\bar{x}_4\bar{x}_5\bar{x}_6x_7 + \ldots$$

Table A.1

Mod 23 Subtractor - Squarer

```
.i 10                      .o 5                       .p 384

inputs x   out y           inputs x   out y           inputs x   out y
9876543210 43210           9876543210 43210           9876543210 43210
---------- -----           ---------- -----           ---------- -----

0000000110 01001           001100-010 10000           -111100000 10000
001010-001 10000           0-01000110 10000           001000-000 10000
0-00100101 10000           00000-1111 10000           0-00000100 10000
0001000-01 00001           00000001-1 00010           0010100-10 00001
001101-001 00100           1-00100110 00100           -101000001 00100
-110100010 00100           00010-1101 00100           -1-1100010 01000
00101-1110 00100           1-10000010 00010           000101-100 00010
000010-000 00001           0-00000001 00001           00110-0011 01000
001010-100 00001           -001100110 01000           00011-0000 01000
00-00-1101 01000           -000000011 01000           -001000101 01000
00101-0010 01000           00110-0001 00010           -011000101 00001
00100-0001 01000           00001-0010 00001           -000100100 01000
-000100110 00010           00101-0110 00001           00101-0000 00010
000101-001 10010           1-00100010 10010           000011-000 10010
1-00000001 10010           0-11000000 01100           00110-1111 01100
00011-1001 00101           -110000011 01100           00001-1010 01100
-111000101 01100           -100100011 00101           00011-1100 01100
-101100101 00101           00101-1011 00101           -100100000 01100
00000-1001 01100           000101-010 00011           -110100100 01100
1-10000000 01001           00110-1001 01001           -101000011 00011
000001-100 01001           00011-1010 00011           1-00000000 00011
00101-1000 01001           00110-1100 01001           -100100110 01001
-011100001 01001           -110000110 01001           00110-1101 00011
-110100110 00011           00001-0111 01001           1--1000011 10000
000111--10 10000           00-011-100 10000           1-10000-01 10000
00-1100-01 00100           000-0000-1 00001           00-0100-11 00100
001-000-11 00001           00-11001-0 00001           000-100-00 00001
0001100--0 00001           0010000--1 00001           1-1001-000 10000
0010-1-000 00100           1-11-00010 01000           000-0-1011 00100
1-0001-100 10000           -1011000-0 00100           000101-11- 01000
1-1-100011 00010           0-110-1010 10000           -10100-110 10000
-1111001-0 00100           -01110-011 10000           1-0-100001 00010
0-101-1001 10000           001100--00 00100           0000-1-001 00001
1-0010000- 00001           -10010-101 10000           1-010000-0 00010
000111-1-1 00010           000001-00- 00001           0--0000110 00100
```

Table A.1

(Continued)

inputs x	out y	inputs x	out y	inputs x	out y
9876543210	43210	9876543210	43210	9876543210	43210
00100-11-1	00100	000011-0-1	00010	0-100-1000	10000
-10000-100	10000	0-011-0111	10000	001-0-1011	00010
1-10-00100	00001	00010-100-	00001	001001-10-	00001
000001-0-0	00010	-100-00010	00001	000-1-1000	00010
-101-00100	00001	00100-101-	00001	-111100-10	01000
00010-1-11	01000	-1011001-0	00010	001-1-1100	00010
-1000000-1	00010	-011-00000	00001	-1-0100010	00010
00100-10-1	00010	-0111000-0	00010	-1100001-1	00010
00010-1-01	00010	000-0-0111	00010	-1-0000101	00001
-11111-010	01000	00010--000	00100	-11011-010	00010
-1011-1000	01000	--00000010	00100	00100--010	00100
1-010-1101	00010	--01000100	00100	-1000-1011	01000
-1111-1100	01000	1-010-0001	00001	-1110-1001	00010
-1100-1001	01000	-00111-100	00001	0-000-1110	01000
-1100-1111	01000	1-100-0011	00001	-1001-1100	01000
-1001-1110	00010	-1101-0000	01000	-0101-1010	00010
0-000-0111	00001	-0000-1101	01000	-1010-0101	00010
-1000-0101	01000	-0111-0100	01000	-1001-0100	00010
-0100-0111	01000	0011-1-010	00110	1-0100011-	00110
-111-00011	00110	001011-00-	00110	-11100001-	00110
1-00-00101	00110	1-1-100001	01001	00011-111-	00110
1-0000010-	00110	0001--1110	00110	-110-00001	00110
-11000000-	00110	1-1-100101	00011	00001-110-	00110
000011-1-1	01001	-0011-1011	10010	0000--1100	00110
-1011-0011	10010	001011-1-1	00011	-0010-1010	10010
-0111-1111	10010	-1010-0010	10010	-1111-0111	10010
-0110-1110	10010	-1110-0110	10010	-0001-1001	10010
-1001-0001	10010	-0000-1000	10010	-0101-1101	10010
-1000-0000	10010	-1101-0101	10010	1-0100-001	01100
0-0111-001	01100	-0100-1100	10010	1-0010-011	01100
-1100-0100	10010	1-1000-011	01100	0-0011-010	01100
1-0010-000	01100	1-0100-100	01100	0-0111-100	01100
0-1001-010	01100	0-1101-100	01100	0-0001-001	01100
1-1000-110	01100	1-000-1010	00101	1-1001-001	01001
-11101-001	01001	1-0011-100	01001	-10101-000	01001
1-010-1111	01001	1-010-1011	00011	-1110-1011	01001
1-001-1110	01001	-1111-1001	01001	1-1-100-10	10000
00-101-1-1	10000	-1011-1110	01001	-10011-000	00011
-1110-1000	01001	-1101-1010	01001	-1010-1101	01001

Table A.1

(Continued)

inputs x	out y
9876543210	43210
-1001-1111	01001
1--1100-00	10000
-1101-0111	01001
1-11-1-010	10000
1-0011-1-1	10000
1-1-100-00	00100
00-1-1-000	01000
1-010-1-10	10000
00-101-11-	00001
1-001-1-01	10000
-1-1-00001	01000
1-000-1-00	10000
-1-1000-00	01000
00-00-1-11	00010
-10101-1-1	00100
-1-101-000	00100
-01111-0-0	00100
-101-1-001	00001
1-1-10-110	00010
-1-111-010	00001
-11001-01-	00001
-11101-10-	00001
-1011--111	10000
-1-101-100	01000
1-10--1110	00001
1--01-1010	00010
-1111-10-0	00010
0-1001--11	00010
1--00-1101	00001
1-000-1-01	00001
-1-01-0011	01000
-0011-1-01	01000
-0010-1-00	01000
--001-1111	00100
--111-1000	00001
1--110-010	01100
1--110-101	01100
1-11-1-000	01001
-01111-01-	00110
-11011-100	00011
00-001--11	10000
-0111-1110	00011
1-1-11-001	10000
1-11-00-01	00100
1--110001-	00001
00-001-1-1	00100
001-01-11-	00010
-11011--01	10000
1-01-00-11	00010
-11001--00	10000
00-01-111-	01000
-1-1100-00	00010
-1-111-001	00100
1-11-1-001	00010
1-1-11-000	00010
1-01--0111	00100
1-0-0-0111	00100
-11-11-000	00001
1-001-101-	00001
--111-1011	10000
0-1101-1-1	00010
1--110-100	00010
1-0-1-1100	00010
-11001-0-1	00010
1-000-11-1	00001
-10101--01	00010
1-100-11-1	00010
-00001-11-	00001
-0-01-1011	01000
-1000-11-1	00010
-0-00-1010	01000
-0100-1-10	01000
-1000--110	00100
-01111-11-	10010
1-11-0-101	01100
0-1011-11-	01100
0-0101--11	01100
0-1011--11	01100
-0111-1010	01001
-1010-0111	01001
-1110-0111	00011
1-0101-11-	10000
00-011-11-	00100
-11101--10	10000
1--1-00010	00001
0001-1--11	00001
1-11-001-0	00010
1-11-00-10	00001
1-00000-1-	01000
0010-1--00	00010
1-1-1-1010	00100
1-1-0-1000	00100
1--111-000	01000
0-0111-01-	00010
1-0011-11-	00010
-11111-11-	00001
1-0001--11	01000
-10111-0-0	00010
1-0001-1-1	00010
1-01--1100	00001
-11111-1-0	00010
1-1000-10-	00010
1-000-10-1	00010
-11-1-1000	00010
-10-0-1111	00010
1-000-011-	01000
-1000-111-	00001
-0111-1101	01101
-1-11-0101	01000
-1-00-0010	01000
--1110-001	00100
--000-1110	00100
1-11--0111	10010
1-1-10-100	01100
1-1-11-010	01001
0-1001-1-1	01100
1-10--1001	00110

Table A.1

(Continued)

inputs x	out y	inputs x	out y	inputs x	out y
9876543210	43210	9876543210	43210	9876543210	43210
----------	-----	----------	-----	----------	-----
1-0101-1-1	01001	1--11-1000	00110	-10011-10-	00110
-100-1-100	00110	1-1-1-1111	01001	-10001--11	00110
1-11--1111	00011	-1-111--11	10000	1--11-1-11	10000
1--111--01	00100	1-0-11-0-0	00001	1-0-01-0-1	00001
1--011--11	00100	1--101--00	00100	1-1-11-1-0	00001
1-1-01-1-1	00001	1--001--10	00100	-111-1-1-1	00001
1--11-110-	00001	1-1-1-1-00	01000	-1-11-1-01	00100
-11-01--11	00010	-10-1-10-0	00001	-110-1--11	00001
-10-0-10-1	00001	-1-01-1-11	00100	1--11-11-0	00010
-1-10-1-00	00100	-01111--0-	01000	-11-1-11-0	00001
-11-0-11-1	00001	-1-00-1-10	00100	1--0--0111	01000
1-100-1-0-	00010	-10-1-1-00	00001	-1100-1--1	00001
1-1-1--111	00100	--1111-1-1	00100	-1-01--111	00100
1-11--101-	00110	-101-1-11-	00110	1--111-11-	01001
1-11-1--11	01001	1-11--1-0-	01000	-1-0-1-11-	01000

Table A.2

Mod 23 Adder

.i 10 .o 5 .p 236

inputs x	outputs y	inputs x	outputs y	inputs x	outputs y
9876543210	43210	9876543210	43210	9876543210	43210
1-0-000-00	10000	00--01-000	10000	1--00000-0	10000
000-01--00	10000	1-00000--0	10000	00-001-0-0	10000
---0000001	00001	--00100-00	00001	--00000-01	00001
00-01--000	00001	00-00--001	00001	--0100000-	00010
00001---00	00001	0001---000	00010	--00-00010	00010
00010--00-	00010	--0000001-	00010	0000---010	00010
000--0-100	00100	0-1-00000-	00100	0-0-00010-	00100
001000-0--	00100	0-10-000-0	00100	0-00-001-0	00100
001-0-100-	01100	000-0-110-	01100	0010--10-0	01100
0000--11-0	01100	-1--0-1000	10000	-10-0-1-00	10000
00000-1---	01000	-1-00-10-0	10000	-1000-1--0	10000
0-01-0-000	00010	0-0100-00-	00010	-1000-1-01	00001
0-00-0-010	00010	0-0000-01-	00010	0--01-1000	00001
0--00-1001	00001	00-11001-1	01000	0-001-1-00	00001
-1110-0001	01000	001-100-11	01000	-1010-0101	01000
-1101-0010	01000	-1001-0110	01000	1-0-01--00	01000
1--001-0-0	01000	-1100000--	01100	-1000001--	01100
00-00-101-	01010	-11-1-11-1	00001	-1---00000	01000
-1-0-0000-	01000	-100-00-0-	01000	1--0-0000-	10000
1-00-00-0-	10000	--0-1000-0	00001	--0-0000-1	00001
00-0-1-00-	10000	0000-1--0-	10000	000---1-00	01000
000-1--0-0	00001	000-0--0-1	00001	00-0--100-	01000
0000--1-0-	01000	00-1-0--00	00010	00-100--0-	00010
0--1-00-00	00010	0--1000-0-	00010	00-0-0--10	00010
0--0-00-10	00010	0--0000-1-	00010	0011-00-1-	01000
00-1-0011-	01000	-10--00100	01100	1--1-00000	10010
-011100--1	01000	-111100--1	10000	001---1000	01100
-1-11001-1	10000	000001--1-	10010	-11-100-11	10000
00-00--11-	00010	1----1-000	01000	00--1-0111	01000
-1110-1-10	00001	-1-10-1110	00001	1-0001----	01000
--1-100000	00101	-111-1--1-	01000	00-11-11-1	10000
-110--1101	00010	-1101-110-	00010	001-1-1-11	10000
00--1-1111	10000	00000--1-1	00101	-1-111--1-	00010
1--1--111-	01000	1--1--1-11	00010	001--001--	01000
-1-0--100-	10000	-100--1-0-	10000	0-0-1-10-0	00001
0-0-0-10-1	00001	-111-00-1-	10000	-1-1-0011-	10000
-11-0-11-0	00001	1--0-1-00-	01000	1-00-1--0-	01000
--1-0--000	00100	-101-1-0-1	00100	-10111-0--	00100
--0-0--100	00100	0011--1-1-	10000	00-1--111-	10000

Table A.2

(Continued)

inputs x	out y	inputs x	out y	inputs x	out y
9876543210	43210	9876543210	43210	9876543210	43210
--100--0-0	00100	-10-11-01-	00100	--000--1-0	00100
1-01--10-1	00100	1-01-1-0-1	00100	1-0---1011	00100
1-0-1-101-	00100	1-0-11-01-	00100	-1--1-0111	10000
-0111-1--1	10000	-1-1-1-11-	01000	--1101--10	00001
-1-1-1--11	00010	1-11-1--1-	10000	1--1-1-11-	10000
1--10--110	00001	--1011-10-	00010	1-11--1-1-	01000
1-10---101	00010	-1111-1--1	00001	1--11-1-1-	00010
-1-11-11-1	00001	-111--11-1	00100	-1111-11--	00100
1--1-1--11	00010	1--111--1-	00010	-11-1-1-11	00001
-1--1-1111	00001	-11---1111	00100	-11-1-111-	00100
-111--1-11	00010	-1111-1-1-	00010	-1-1--1111	00010
-1-11-111-	00010	--1-11-1-1	00001	1-1-1--1-1	00001
-10--000--	01000	1-0--000--	10000	000--1-0--	10000
00--10---0	00001	00--00---1	00001	000---10--	01000
0---100--0	00001	0---000--1	00001	-11--001--	10000
-1-0-1--01	00010	-1-011--0-	00010	001---11--	10000
1--0--1-01	00010	1--01-1-0-	00010	1--0-1--01	00010
1--011--0-	00010	-10--1--11	00100	--1-01-1-0	00001
1-1-0--1-0	00001	1--11-10--	00100	1--111-0--	00100
1-0--1--11	00100	-11111----	01000	-1-111-1--	01000
-11--1--11	01000	1-11-1---1	10000	1--1-1-1-1	10000
1--111-1--	10000	--011--0-1	00100	---101-11-	00001
1-1--1--11	10000	1-1-11--1-	10000	1---11-11-	10000
1-11--1--1	01000	--0-1--011	00100	--10-1-1-1	00010
1--1--11-1	01000	1--11-11--	01000	1-11----10	00001
1-1---1-11	01000	1-1-1-1-1-	01000	1-1-1--10-	00010
1-----1111	01000	1---1-111-	01000	--1111---1	00001
---111-1-1	00001	--11-1-1-1	00100	--1111-1--	00100
--1-11--11	00001	1--11--1-1	00001	1-11---1-1	00100
--1-11-11-	00100	1-1-1---11	00001	--11-1--11	00010
--1111--1-	00010	1---1--111	00001	---111-11-	00010
1-1----111	00100	1-1-1--11-	00100	1-11----11	00010
1--1---111	00010	1--11--11-	00010	--111--1-1	00100
--1-1--111	00100	-1--01---0	00001	-10---10--	10000
1---0-1--0	00001	1---01---0	00001	1-0--1-0--	01000
---10---00	00010	---00---10	00010	--10---00-	00100
--00---10-	00100	-1--11---1	00001	-11--1-1--	01000
---01---01	00010	1---1-1--1	00001	1---11---1	00001
1-1--1-1--	10000	1-1---11--	01000	--01---01-	00100
---11---11	00010	--11---11-	00100		

Table A.3

Mod 11 Subtractor - Squarer

.i 8 .o 4 .p 78

inputs x	out y	inputs x	out y	inputs x	out y
76543210	3210	76543210	3210	76543210	3210
-1010000	0010	0000-101	0010	-1110010	0010
0010-111	0010	0001-100	1000	-1000001	1000
-1010010	1000	0010-101	1000	-0110000	1000
0000-011	1000	-1110QQ0	0100	0000-111	0100
00100-00	0100	0-000010	0100	1-000000	1001
00001-00	1001	00011-00	0101	1-000001	0101
1-1-0010	1000	00101-1-	1000	-111-100	1000
-100-111	1000	-011-110	1000	-110-011	1000
-0110-01	0100	0-01-011	0100	0-10-100	0100
-1000-10	0100	001-1-00	0011	1-00001-	0011
-11-0001	0011	0001-11-	0011	000--110	0011
-110000-	0011	1--10001	1001	00011--1	1001
-1011-00	1001	1-00-101	1001	1-1--111	1000
-1111-1-	1000	00-11-1-	0100	1-1-00-1	0100
1--1-110	1000	-1101--1	1000	1--100-0	0100
00-01--1	0100	-101-001	0101	-001-101	0101
1-1-1-00	0100	1-001-1-	0100	-100-000	0101
-000-100	0101	-111-011	0101	-011-111	0101
-110-010	0101	-010-110	0101	1-00-1-0	0100
-1101--0	0100	--11-101	0100	-101--11	0100
1-0--011	0011	-0111-0-	0011	1--1-100	0011
-1001--1	0011	1--1-1-1	0100	-1-11--1	0100
1--11--0	0001	1--01--1	0001	0--01-1-	0001
1-1-0--0	0001	0--10--0	0001	0--00--1	0001
---10-10	0001	1-1--10-	0011	-10-1-1-	0011
0-10---1	0001	---0--11	0001	--11---0	0001

Table A.4

Mod 11 Adder

.i 8 .o 4 .p 79

inputs x 76543210	outputs y 3210
1-0000-0	1000
0001--00	0001
000--110	0110
1---0000	1000
0000---1	0001
---10000	0001
0-000-01	0001
0000--1-	0010
-110-001	0100
-110-110	0001
1-0-000-	1000
000--10-	0100
0--100-0	0001
001--11-	1000
-11-1-1-	0100
-1-11--1	0001
-11--111	0010
-111-1-1	0001
-1-01--0	0001
-10-1--1	0010
1-1-1-1-	1000
--10--00	0010
1----111	0100
1-1--1-1	0100
--111-1-	0010
1--1--11	0001
--11--11	0010
00-01-00	1000
0-0-0010	0010
0010-10-	0110
-100-1-0	1000
-1-0-100	1000
00---100	0100
0-010-00	0001
00-1-111	1000
-01100-1	0100
-100001-	0110
000-1-0-	1000
0--000-1	0001
00-10--0	0001
-1011-0-	0010
1-0--101	0010
1-1--11-	0100
-111-11-	0010
1-001--0	0101
-10--10-	1000
1--11-1-	1000
1--1-10-	0010
-1111---	0100
1--1-11-	0100
1-1---10	0001
1-1---11	0010
1-0-1--1	0110
--000001	0001
0-10000-	0010
00001---	1000
001-001-	0100
0000-1--	0100
-11100-1	1000
-01-0-00	0010
-101-010	0100
00-1-011	0100
-11-0000	0110
-10-000-	0100
00-00--1	0001
-11-001-	1000
-1-1-011	1000
-011-1-1	1000
1--1-1-1	0001
-1-1-111	0001
1-1-1-00	0101
1--0-1-0	0001
1-1-1--1	1000
--00--10	0010
1--11--1	0001
--101-1-	0001
--01--01	0010
--111--1	0001
1--11-0-	0110

Table A.5

Quantizer

.i 10 .o 1 .p 32

inputs x	y	inputs x	y	inputs x	y
9876543210	0	9876543210	0	9876543210	0
0---0-1111	1	0-0-0-1-11	1	0--00-11-1	1
0-0-0-1-11	1	0-000-1--1	1	1-1--0-000	1
00--0--111	1	1-11-0-00-	1	1-1-10-00-	1
1-11-0-0-0	1	0--0--111-	1	0-00--1-1-	1
1----00-00	1	1--1-00-0-	1	-00-0-0011	1
-0100-01-1	1	1---100-0-	1	-0000-00-1	1
1--1-00--0	1	1--1100---	1	--0--1-1--	1
0-0---11--	1	-010--011-	1	-000--001-	1
000----1--	1	1-1--00---	1	0----1----	1
-1111-00--	1	00----1---	1	-111--000-	1
-111--00-0	1	-11-1-000-	1	-11---0000	1

Table A.6

Mod 23 Minimum Computer

```
.i 10             .o 1              .p 26

inputs x    y     inputs x    y     inputs x    y
9876543210 0      9876543210 0      9876543210 0
---------- -      ---------- -      ---------- -

--111-0--- 1      -0--11--1- 1      0--111---- 1
0---1-110- 1      ---0---11- 1      1-0-0---0- 1
--0-1-1-1- 1      1--1--01-- 1      -0--01--0- 1
0---0-1-1- 1      --0-1-010- 1      -1--00--0- 1
-1--10--1- 1      0-1--1---- 1      -0-110--0- 1
0--10--0-- 1      1-1---1--- 1      --0-0-0-1- 1
---01--00- 1      1--0-00--- 1      0-0---1--- 1
---00--1-- 1      ---1---01- 1      001---0--- 1
----1----0 1      ----0----1 1
```

Section 4 discussed PLA implementation of the reduced functions of tables A.1 through A.6. Considerable reduction in the hardware is possible if random logic is used instead. To illustrate, figure A.1 shows a mod 23 adder layout obtained with special purpose CAD tools that were developed by MITRE's integrated electronics project staff. The silicon area required by the adder is 228λ × 133λ which is more than 20 times smaller than the area of the PLA given in section 4. Similar reduction can probably be obtained with the other DTW cell functions, so the estimate of the area of one cell given in section 4 is quite pessimistic.

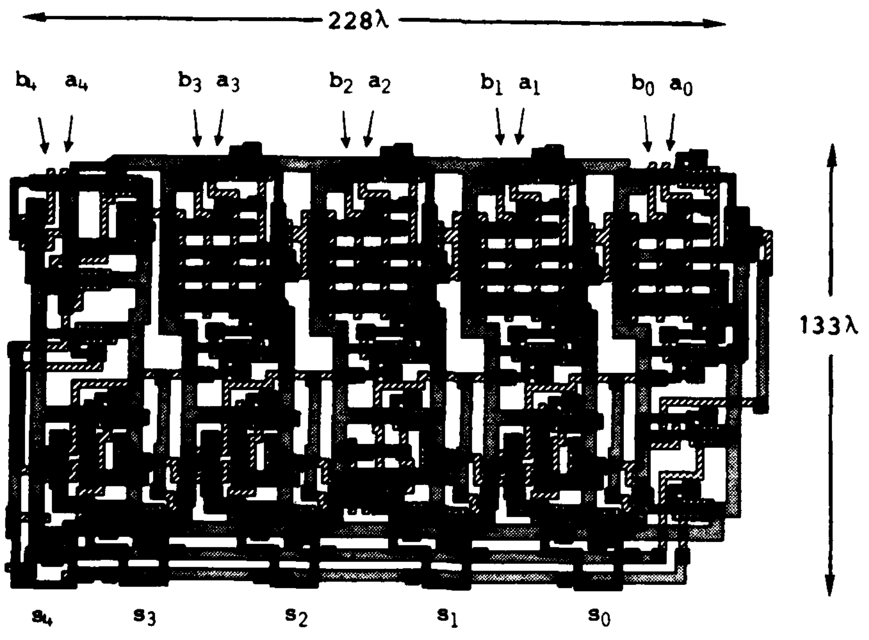


Figure A.1. Mod 23 Adder

Table A.7 shows a two-input gate count of a combinational logic (sum-of-products) implementation of the six expressions.

Table A.7

Number of Two-Input Gates Required by a Sum-of-Products Implementation of the DTW Cell Functions of Table 4.1

Function	Inverters	ANDs	ORs
Mod 23 Subtractor-Squarer	10	2945	510
Mod 23 Adder	10	1360	248
Mod 11 Subtractor-Squarer	8	437	105
Mod 11 Adder	8	385	86
Quantizer	10	96	25
Mod 23 Minimum Computer	10	171	31
Total		5394	1005

DISTRIBUTION LIST

INTERNAL

A-10

D.M. Levine
G. MacDonald
A.J. Tachmindji

D-10

E.L. Key

D-11

R.D. Haggarty
B.M. Horowitz
J.K. De Rosa

D-14

G.F. Steeg

D-80

H.B. Goldberg
R.W. Jacobus
D.A. Kahn
J.D.R. Kramer
F.K. Manasse
S.M. Newman

D-82

A. Bark
C.J. Beanland
S.S. Benkley

D-82 (Continued)

A.L. Bequillard (5)
T.J. Brando
M.F. Bridgland
D.R. Bungard
D.O. Carhoun (5)
A.H. Chan
H.E.T. Connell
R.J. Cosentino
J.H. Cozzens
S. Dhar
R.C. DiPietro
G.A. Doodlesack
W.L. Eastman (5)
J.C. Fohlin
R.A. Games
G.L. Glatfelter
M.A. Gray
J.D. Harris
B.L. Johnson
A.K. Kamal
M. Kaplan
V. Kross
R.V. Labonte
L. Lau
J. Low
H.H. Ma
J.P. Manosh
T.J. McDonald
R.C. McLeman (10)
S.J. Meehan
D. Moulin
D.J. Muder
C.L. Nowacki
S.D. O'Neil

DISTRIBUTION LIST (Concluded)

D-82 (Concluded)

E.A. Palo (5)
B.D. Perry
M.A. Poole
V.K. Proulx
R. Rifkin
J.J. Sawyer
M.A. Schuster
W.K. Talley
J.J. Vaccaro
C.J. Valas
M.R. Weiss

D-86

B.N. Suresh Babu

D-97

A.P. Doohovskoy
D.H. Friedman
W.D. Hall
C.E. Reid
J.E. Roberts

W-35

D.J. Jurenko

W-37

G. Benke

W-86

J.C. Blake

PROJECT

Rome Air Development Center
Electronic Systems Division
Air Force Systems Command
Griffiss AFB, New York 13441

OCTS

J. Graneiro
S. Lis
Z. Pryk (10)
C. LaViera

MISSION
of
Rome Air Development Center

END

12-86

DTIC